Universidade Nova de Lisboa

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Electrotécnica

Arquitectura Orientada a Serviços REST para Laboratórios Remotos

Dissertação para obtenção do grau de

Mestre em Engenharia Electrotécnica e de Computadores

Márcio António Fernandes Ngolo

Lisboa, Julho - 2009

Orientador: Prof. Doutor Luís Brito Palma

Departamento de Engenharia Electrotécnica

Da Faculdade de Ciências de Tecnologia

Da Universidade Nova de Lisboa

Agradecimentos

Ao Professor Luís Brito Palma, na qualidade de orientador científico, agradeço a oportunidade concedida para a realização deste trabalho, a dedicação e o profissionalismo demonstrado, assim como a amizade e motivação, sem deixar de salientar a enorme disponibilidade apresentada para resolução dos problemas inerentes a realização da tese, principalmente nos momentos de maiores dúvidas, sem duvida o meu grande mentor neste último período do curso.

Aos colegas do laboratório 2.6 pela colaboração, compreensão e partilha de conhecimentos, durante o tempo de realização da tese. Quero nomeadamente agradecer ao Fernando Mapota, ao João Meireles, ao Ricardo Fernandes, ao Tiago Oliveira e ao Gonçalo Cabecinhas.

Aos meus colegas de curso, pela colaboração, amizade e solidariedade tão importantes na difícil vida académica.

À Dra. Maria João Moutinho pelo apoio e pelos ensinamentos que me transmitiu durante os mais importantes períodos da minha vida académica e pessoal.

Por fim quero agradecer de forma especial, à minha mãe Catarina e ao meu pai Eugénio pelo apoio e amor incondicional que sempre me dispensaram, sem os quais não teria sido possível realizar a minha formação.

Resumo

A principal contribuição apresentada nesta dissertação é uma arquitectura orientada a serviços REST, que permite a implementação de um laboratório remoto com acesso via Internet, utilizando um cliente leve.

O principal objectivo foi o desenvolvimento de um cliente leve baseado em tecnologias AJAX (Asyncronous Javascript and XML) executado em navegador (browser) de Internet, com funcionalidades de comunicação com uma aplicação que corre num servidor. No lado do servidor foi utilizada uma aplicação de instrumentação virtual suportada pelo Labview (versão 8.6), que permite controlar processos através de placas de aquisição de dados. Ambas as aplicações correm no servidor e aproveitam as potencialidades das arquitecturas orientadas a serviços, especificamente usando Serviços Web baseados na arquitectura REST. A utilização do cliente leve proposto permite ultrapassar a necessidade de ter instalado um software do tipo "Labview Pluaggin" no navegador do cliente.

O paradigma da arquitectura orientada a serviços permite que cada operação do laboratório remoto possa ser ela própria um serviço, o que permite a criação de serviços distribuídos em sistemas de controlo.

Palavras-chave: Sistemas Remotos, Serviços Web, Programação Web, Controlo de Processos, Laboratórios Remotos.

Extended Abstract

The main contribution in this dissertation is to present a REST services oriented architecture to the implementation of a remote lab accessible using Internet with a thin client.

The main objective was the development of a thin client based on AJAX (Asynchronous JavaScript and XML) technologies wich is executed in a browser and that communicate with a server side application. In the server side a virtual instrumentation application supported by Labview (version 8.6) was used. This application is used for process control using data acquisition boards. Bought applications runs in the server and takes advantage of the service oriented architectures, specifically Web Services using REST design style or simply REST services. The use of a thin client allowed the need to use any kind of software in the client side like a pluggin in the client browser.

Keywords: Remote Systems, Web Services, Web Programming, Process Control, Web Lab.

Índice Geral

Simbologia

AJAX: Asynchronous JavaScript and XML

API: Application Programming Interface

BPEL: Business Process Execution Language

CSS: Cascading Style Sheets

CPLD: Complex Programmable Logic Device

CORBA: Common Object Request Broker Architecture

DAQ: Data Acquisition

DOM: Document Object Model

DCOM: Distributed Component Object Model

DHTML: Dynamic HTML

FPGA: Field Programmable Gate Array

HTTP: Hypertext Transfer Protocol

HTML: Hypertext Markup Language

JRE: Java Runtime Environment

J2ME: Java Platform Micro Edition

MIT: Massachusetts Institute of Technology

NI: National Instruments

REST: Representational State Transfer

RTE: Real Time Systems

ROA: Resource Oriented Architecture

SOA: Service Oriented Architecture

SOAP: Simple Object Access Protocol

SLA: Service Level Agreement

SVG: Scalable Vector Graphics

QOS: Quality of Service

UDDI: Universal Description and Discovery Integration

URL: Uniform Resource Locator

VI: Virtual Instrumentation

VISA: Virtual Instrument Software Architecture

XML: Extensible Stylesheet Language

XLST: Extensible Stylesheet Language Transformation

W3C: World Wide Web Consortium

WSDL: Web Service Definition Language

Lista de Tabelas

Lista de Figuras

1. Introdução

1.1 Motivação

A forma como a informação é disponibilizada é um factor crítico actualmente, muitas vezes existe a dificuldade em colocar a informação no formato correcto ou utilizar os meios mais adequados para que esta possa ser utilizada da melhor maneira possível.

Na área de investigação relacionada com Controlo e Decisão muitas vezes encontram-se dificuldades, quando pretende-se apresentar informação relativa a processos de controlo através da Internet. Tal deve-se a problemas relacionados com incompatibilidades entre o software que se utiliza para representar determinado sistema e o software que se encontra disponível do lado do cliente. Os Serviços Web oferecem formas mais transparentes de se ter acesso a esses sistemas remotamente. Estas novas formas de se aceder a informação, podem ser aproveitadas para a criação de soluções de comunicação com maior grau de compatibilidade de forma a serem usadas para a implementação de laboratórios remotos. Esta solução simplifica o acesso a soluções deste tipo.

A criação de um laboratório remoto usando uma arquitectura orientada a serviços é a principal motivação para a realização desta tese.

A implementação de um cliente leve do lado cliente de modo a que não haja perda de eficiência no browser do cliente é também um objectivo importante, como se refere de seguida

1.2 Objectivos Principais e Contribuições

Actualmente no ensino da engenharia de um modo geral, e no ensino da engenharia de controlo de um modo particular, é cada vez mais necessário o conceito de laboratório remoto ou sistema remoto. Na indústria começa a existir a necessidade de tornar os projectos de sistemas de controlo acessíveis pela Internet, de uma maneira fácil e transparente. De facto existe a necessidade de se poder disponibilizar meios laboratoriais e industriais de forma remota (Coito & Brito Palma, PETRA 2008), de modo a tornar menos condicionado o acesso a processos e equipamentos.

Os principais contributos desta tese são os seguintes:

- A Criação de uma arquitectura e plataforma de software para suportar um ou mais laboratórios remotos, que permita o acesso ao laboratório apenas com a utilização de um navegador Web ou browser.
- O desenvolvimento de um cliente leve a partir do browser, baseado em tecnologias AJAX, sem a necessidade de software adicional ("Labview Pluggin ou outro);
- Tornar a experiencia do utilizador mais interessante com a utilização do paradigma Web 2.0, que é um paradigma que pretende aproximar a experiencia Web da experiencia que o utilizador obtêm com aplicações desktop, nomeadamente com o uso de tecnologia como AJAX, serviços Web entre outras.
- Criação de uma plataforma para laboratório remoto orientada a serviços Web, o que facilita a adição de novas funcionalidades e de novos serviços (experiencias remotas), o que torna a plataforma não só escalável mas também distribuída.

1.3 Organização da Tese

Está Tese encontra-se organizada da forma descrita seguidamente.

No capítulo 1 Faz-se uma introdução à tese e apresenta-se a estrutura e organização do trabalho.

No capítulo 2 realiza-se um levantamento geral sobre o estado da arte na área dos laboratórios remotos, destacando-se as principais arquitecturas e tecnologias. São referidas as principais implementações de arquitecturas orientadas a serviços e feita uma particularização sobre os Serviços Web. As soluções em termos de software usado na implementação de aplicações cliente e servidor são igualmente apresentadas.

No capítulo 3 é apresentada a arquitectura proposta assim como a solução e os meios usados na obtenção da mesma. São explicados pormenorizadamente, as opções tomadas para a realização da arquitectura. É também apresentado e explicado o código Labview utilizado para programar a aplicação servidora.

No capítulo 4 faz-se uma análise global do trabalho e são apresentadas as conclusões finais do trabalho. São igualmente propostas melhorias técnicas e sugestões para trabalho futuro.

Anexo A – É apresentado um resumo da placa de aquisição NI-USB 6009 utilizada no trabalho

Anexo B – É apresentado o código javascript utilizado para programar a aplicação cliente

Referências Bibliográficas – São apresentadas as referências utilizadas durante a realização do trabalho, nomeadamente, livros, artigos e páginas Web.

2. Estado da Arte

2.1 Introdução

Neste capítulo serão apresentadas definições e discutidas algumas tecnologias relacionadas usadas para a implementação de laboratórios remotos.

Os serviços Web são igualmente analisados, sendo feitas referências às principais tecnologias usadas para a sua implementação. Para finalizar a apresentação de alguns exemplos de laboratórios remotos implementados em universidades portuguesas e estrangeiras.

2.2 Tecnologias de Suporte para Laboratórios Remotos

2.2.1 Introdução

A necessidade de se utilizar a Internet como meio de difusão privilegiado para experiências laboratoriais, tem aumentado o desafio daqueles que pretendem tirar partido desse importante meio. Dois dos mais conhecidos fabricantes de software para computação científica, a National Instruments e a Mathworks, têm aproximado os seus produtos do meio Internet. As novas soluções que a National Instruments oferece aos seus clientes, permitem tirar partido do novo paradigma que são os Serviços Web.

As soluções baseadas em Serviços Web permitem que as aplicações clientes sejam totalmente independentes da aplicação servidora, melhorando a flexibilidade do lado do cliente e a partilha de recursos no sistema global.

2.2.2 Soluções de Software para Laboratórios Remotos

A chegada das tecnologias Web 2.0 abriram uma nova janela de possibilidades para os laboratórios remotos. Tecnologias como AJAX, Flash e Java Aplets, permitem que actualmente se possa oferecer aos utilizadores experiencias laboratoriais com uma forte componente interactiva.

Cada vez se desenvolve mais soluções para laboratórios remotos não intrusivas do ponto de vista da segurança, solução para laboratórios remotos que utilizem AJAX, HTML, Flash, Java principalmente, permitindo que o utilizador do laboratório remoto esteja seguro em relação ao uso dos recursos que lhe são postos a disposição. Como não há lugar a transferência de software directamente do servidor onde está instalado para o computador do cliente, o perigo de danos ou de violação de privacidade não existe.

Ao projectarmos laboratórios remotos é importante ter em linha de conta duas questões:

- Que software usar para o cliente?
- Que software usar para servidor?

Respondendo primeiro à primeira questão, existem diversas tecnologias para o desenvolvimento do cliente. Tenha-se em linha de conta as seguintes tecnologias analisando-se seguidamente os seus prós e contras e particularmente a razão da escolha para esta tese de uma destas tecnologias.

Tecnologias para Suporte

ActiveX: O ActiveX é uma poderosa ferramenta para o desenvolvimento de aplicações Web, no entanto tem duas grandes desvantagens em relação a outras soluções. A primeira é de que apenas o Internet Explorer executa código ActiveX e o segundo é que as aplicações ActiveX são intrusivas (precisam de aceder a ficheiros do sistema), o que

muitas vezes inviabiliza a utilização de sistemas Web que utilizam essa tecnologia por falta de privilégios dos utilizadores nas máquinas que utilizam.

Java Applets: As applets Java permitem desenvolver aplicações Web muito ricas em termos de recursos gráficos, caso o Java Runtime Environment (JRE). Apesar de o JRE estar já instalado em muitos sistemas Web, o JRE apresenta um problema principal que é a existência de muitas versões, e o facto das diferentes versões não serem retro-compatíveis. Por outro lado devido a diminuição da popularidade das applets Java, o JRE não se encontra instalado em muitos sistemas, o que cria o problema por exemplo da falta de privilégios nos sistemas em que utilizador não seja o proprietário da máquina ou computador de teste.

Labview Pluggin. O runtime Labview pluggin é uma aplicação que permite a visualização e a interacção de aplicações Labview via Internet de forma rápida. No entanto da parte do utilizador é requerida a instalação do pluggin Labview com um tamanho de cerca de 70MB, não sendo nos tempos actuais um ficheiro grande tendo em conta a qualidade da Internet na Europa, não é porém uma experiência optimizada uma vez que em termos globais dependendo da aplicação Labview, entretanto desenvolvida podemos ter uma ocupação de largura de banda que poderá ser elevada.

Adobe Flash. O adobe flash é uma das ferramentas mais comuns e mais usadas para a criação de experiências Web com forte componente gráfica, podendo até incluir vídeos de acesso a Serviços Web, entre outras experiências. No entanto o adobe flash não partilha uma solução comum para todas as plataformas, por exemplo a versão para Linux encontra-se desfasada da versão Windows em pelo menos duas gerações sendo que a versão Linux é de instalação mais problemática do que a versão Windows. Por outro lado a instalação do adobe flash no sistema do utilizador é invasiva apesar das aplicações flash não o serem. Outros dos problemas do adobe flash está no facto de ser um sistema proprietário estando o seu desenvolvimento dependente da empresa que o desenvolve, neste caso a adobe. Esta situação pode colocar problemas de segurança no caso em que as falhas de segurança do software são expostas por hackers (que tendem a atacar sistemas proprietários).

AJAX. O AJAX é uma solução que se apresenta comparativamente com as soluções já apresentadas como aquela que oferece mais garantias para a implementação de laboratórios remotos. AJAX significa (Assyncronous Javascript and XML), não é propriamente uma tecnologia nova. O AJAX é uma combinação de tecnologias já bem conhecidas como o XML, o Javascritpt, o XML, o CSS e o DOM, com adição de um objecto o XMLHTTPRequest para permitir desenvolver aplicações em que a experiência do utilizador seja bastante enriquecida, sem comprometer a eficiência do sistema do utilizador. Através do objecto XMLHTTPRequest o AJAX chama Serviços Web de forma assíncrona, o que permite actualizar de forma dinâmica dados sem comprometer a eficiência do cliente (browser). O facto de o AJAX usar normas já conhecidos pelos browsers, e o facto de o XMLHTTPRequest já ser um objecto nativo na maioria dos browsers mais actuais, permite olhar para o AJAX como a melhor solução para o desenvolvimento de laboratórios remotos. As capacidades do AJAX permitem mesmo implementar um laboratório remoto totalmente em AJAX do lado do cliente. O utilizador necessita apenas do browser para utilizar um laboratório remoto desenvolvido em AJAX, não precisando de ter privilégios na máquina servidora. A prova de que o AJAX está de facto a ser fortemente usado para o desenvolvimento de aplicações Web é o facto das grandes empresas de Internet como a Google e a Yahoo estarem a utilizar fortemente as capacidades desta tecnologia para melhorar os seus produtos. A Google tem usado esta solução para o caso do seu sistema Google Documents no qual dois utilizadores podem alterar simultaneamente o mesmo documento, sendo que o documento actualiza-se na hora para os dois utilizadores. A tabela 2.1 faz uma breve comparação entre as tecnologias referidas nos parágrafos anteriores.

	ActiveX	**Applets Java**	**Adobe Flash**	**AJAX**	**Labview Pluggin**
Necessidade de Instalação	S	S	S	N	S

Retro Compatibilidade	-	N	N	-	N
Largura de Banda	Alta	Alta	Alta	Baixa	Alta
Áudio e Vídeo	BOM	BOM	Muito BOM	Baixa	-
Multi-Plataforma	Baixo	Médio	Médio	Alta	Baixo
Aceitação pelos Browsers	IE apenas	Média	Média	Alta	Média

Tabela 2.1. Quadro comparativo Tecnologias Web para Laboratórios Remotos (Gomes, Luís & Javier Garcia Zubia,)

Tecnologias para Suporte da Aplicação Servidora

Depois de se analisar as diferentes tecnologias para o desenvolvimento de aplicações cliente para laboratórios remotos, e tendo em conta que aplicação central escolhida para ser executado no servidor é o Labview, importa agora analisar que soluções existem para o desenvolvimento de aplicações servidor.

Uma das razões pela qual foram escolhidos os Serviços Web para o desenvolvimento da aplicação servidor foi a elevada Interoperabilidade dos Serviços Web que permitem uma grande liberdade de escolha no desenvolvimento da aplicação cliente. Com Serviços Web pode-se desenvolver a aplicação cliente em qualquer linguagem de programação ou tecnologia desde que a mesma consiga trabalhar com código XML. A tabela 2.2 mostra uma comparação entre algumas tecnologias para implementação de

aplicações servidoras. A comparação é feita fundamentalmente para explicitar as vantagens do uso de Serviços Web para o desenvolvimento de laboratórios remotos.

	Segurança	**Necessidade de Instalação de Pluggins no lado do Cliente**	**Interoperatividade**
Labview Web Server	Boa	Labview Runtime Engine	Nenhuma
Serviços Web	Suficiente	Não	Total
DataSockets	Muito boa	Labview	Nenhuma

Tabela 2.2. Comparativo entre Tecnologias para implementação de aplicações Servidor para Laboratórios Remotos

Como se vê pela tabela 2.2 (Baccigalupi, et al, 2006), os serviços Web tendo em conta as outras soluções estudadas é aquela que apresenta características mais interessantes para implementação de laboratórios remotos uma vez que é a única que de raiz suporta interoperabilidade. Esta característica dos Serviços Web não só permite uma grande liberdade na escolha da tecnologia para desenvolvimento da aplicação cliente, como permite que os serviços criados possam facilmente ser escaláveis, no sentido em que outros serviços possam ser desenvolvidos por cima dos serviços já criados. por aplicações diferentes localizadas em qualquer servidor remoto. Por exemplo um determinado Laboratório Remoto poderia oferecer um determinado serviço, serviço este que poderia ser aproveitado por outra aplicação localizada remotamente, para a criação de um segundo serviço mais complexo ou serviço composto e oferecer esse serviço composto aos seus clientes.

2.2.3 Instrumentação Virtual

A instrumentação virtual é um paradigma indispensável e usado no desenvolvimento de aplicações para laboratórios remotos. Entende-se como instrumentação virtual a capacidade de por meio de software emular instrumentos virtuais (aparelhos de medida, de visualização etc) para o utilizador possa trabalhar com o mínimo de instrumentos reais. A utilização de software baseado no paradigma da instrumentação virtual, como o Labview possibilita e facilita o desenvolvimento de soluções para a implementação de laboratórios remotos. Uma das principais vantagens da instrumentação virtual é a virtualização dos instrumentos o que permite simular com grande exactidão uma gama variada de instrumentos reais. A emulação de instrumentos de medida como osciloscópios, voltímetros etc, permite uma grande redução de custos. Existe também uma redução do esforço em termos de trabalho, uma vez que uma grande parte dos aparelhos de medida estará emulada no software. Para a equipa que desenvolve a aplicação servidora, isso significa ganhos em termos de produtividade e eficiência. O Labview é em linhas gerais o padrão em termos de software de instrumentação virtual, possuindo a capacidade de realizar, a aquisição de dados, o processamento dos sinais obtidos e a sua representação. Na sua última versão, este potente software permite a publicação das aplicações nele desenvolvidas como serviços Web (RESTful Services), o que vem tornar este software como o ideal para a implementação de laboratórios remotos, uma vez que o suporte aos serviços Web, facilita bastante o trabalho o desenvolvimento global do sistema. A estratégia da National Instruments em oferecer as capacidades dos Serviços Web na sua nova suite (versão 8.6) abre novas possibilidades para os investigadores na área dos laboratórios remotos. Com está solução a maior preocupação passa a ser fundamentalmente a aplicação cliente.

O emergir da instrumentação virtual mudou drasticamente o panorama dos sistemas de medidas automática e dos sistemas de monitorização remota.

Os sistemas actuais de instrumentação virtual têm neste momento a capacidade para acomodar simultaneamente instrumentos com interface, GPIB, série, USB entre outras.

Na figura 2.1 (S.Sumathi, P.Sureka, *Labview Based Advanced Instrumentation and Control*) está representada genericamente a plataforma de instrumentação virtual para aquisição de dados para medidas.

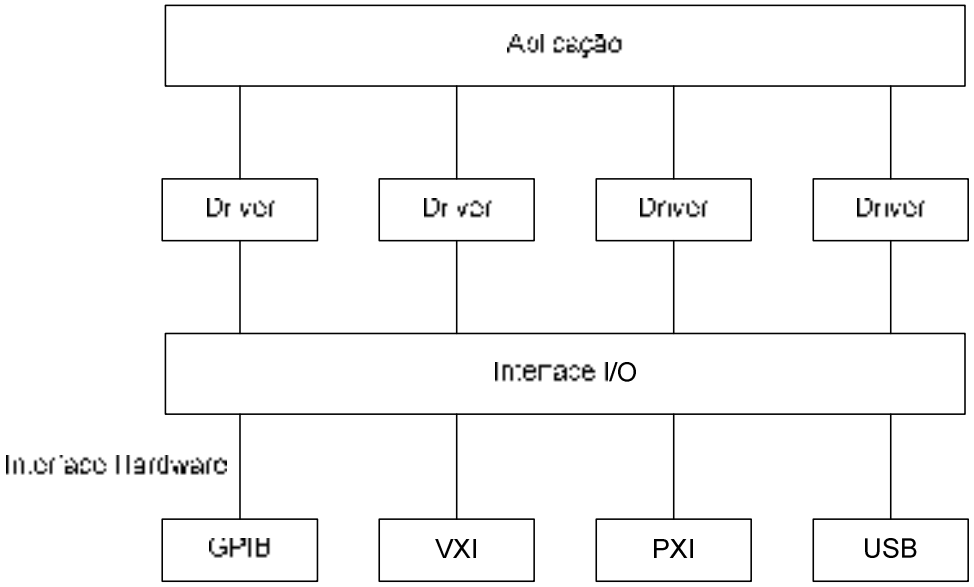

Figura 2.1. Plataforma de Medidas para Instrumentação Virtual para medição

Software Labview

O software Labview segue a arquitectura descrita na figura 2.1. Pode-se então deduzir que um sistema de instrumentação virtual pode ser dividido em drivers (para os instrumentos), interface hardware, interface I/O e aplicação. As aplicações de instrumentação virtual denominam-se por VI(Virtual Instrumentation) , podendo ter também a designação de objecto VI. Os objectos VI estão divididos em três partes: interface de utilizador, terminal de saída e terminal de entrada.

A figura 2.2 (S.Sumathi, P.Sureka, *Labview Based Advanced Instrumentation and Control*) representa um objecto VI com terminais de entrada. A principal vantagem da programação visual, consiste no facto de que a utilização de componentes visuais que

simulam instrumentos físicos é bastante intuitiva para engenheiros e cientistas. Essa qualidade, facilita grandemente o trabalho de desenvolvimento de aplicações, utilizando software de instrumentação virtual, como por exemplo o Labview.

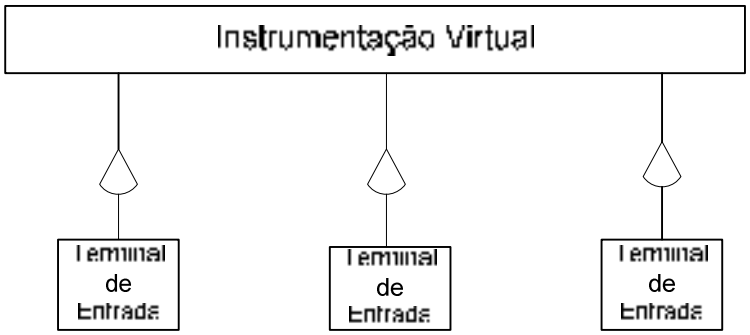

Figura 2.2. Objecto VI

O desenvolvimento da instrumentação virtual requer a utilização de uma interface única para, foi assim criada a interface I/O VISA (Virtual Instrumentation Software Architecture) que é comum a qualquer sistema de instrumentação virtual, e que uniformizou a interface entre os sistemas de aquisição e os sistemas de instrumentação visual. Devido a grande interoperabilidade (independente do sistema operativo e da interface hardware do aparelho de medida) do VISA, a instrumentação virtual ganhou uma uniformização da interface I/O sem a necessidade de mudar sempre que a interface hardware do aparelho é diferente. O VISA permite unificar a interface de hardware o que permite ter uma plataforma comum de aquisição independente da interface do aparelho. O VISA tem as seguintes características comparativamente a outras soluções de software para a interface I/O:

- Funções de controlo compatíveis com diversos instrumentos que incluem os seguintes tipos VXI, PXI, GPIB, USB, RS-232 entre outras.
- As funções de controlo I/O do VISA funcionam em sistemas com multi ou mono processamento de controlo I/O do VISA suportam qualquer mecanismo de rede.

Devido a grande compatibilidade e interoperabilidade (independente do sistema operativo) o VISA permite que novos aparelhos virtuais possam ser integrados num

sistema de instrumentação virtual sem que sejam necessárias grandes alterações ao nível de software.

2.2.4 Programação Visual

As linguagens de programação visual são compostas por nós e por blocos de dados que se interligam entre si, para dar origem a uma determinada aplicação. Por norma, o programador deve ter conhecimento prévio do funcionamento ao nível de tipos de dados e alguns conhecimentos de lógica para poder iniciar uma aplicação em linguagem visual. Um dos principais softwares usado para desenvolver aplicações de instrumentação virtual é o Labview. O Labview suporta os principais sistemas operativos, Windows, Linux, Mac OS. O Labview é uma suite especialmente desenhada para aquisição de dados, controlo de equipamentos, análise e apresentação gráfica de dados. O ambiente de trabalho do Labview permite analisar e apresentar dados de forma interactiva e bastante intuitiva.

A figura 2.4 ilustra o diagrama de blocos de uma aplicação que simula formas de onda. O utilizador pode definir através da interface gráfica, as características dos sinais que pretende visualizar. O utilizador pode definir parâmetros como Amplitude, Frequência, ruído e tipo de sinal.

A figura 2.3 ilustra o painel frontal ou interface gráfica, é essa interface que vai ser mostrada ao utilizador. Está aplicação particular encontra-se já preparada para ser exportada como um Serviço Web, os blocos para serviços Web(HTTP responde mime.vi, Write response.vi, flush output .vi) permite que o sinal possa ser visualizado com recurso a um browser.

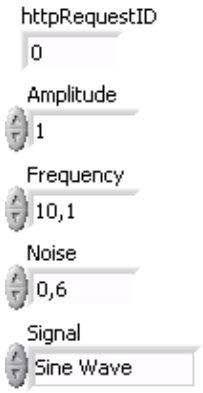

Figura 2.3. Exemplo da Linguagem Visual do Labview

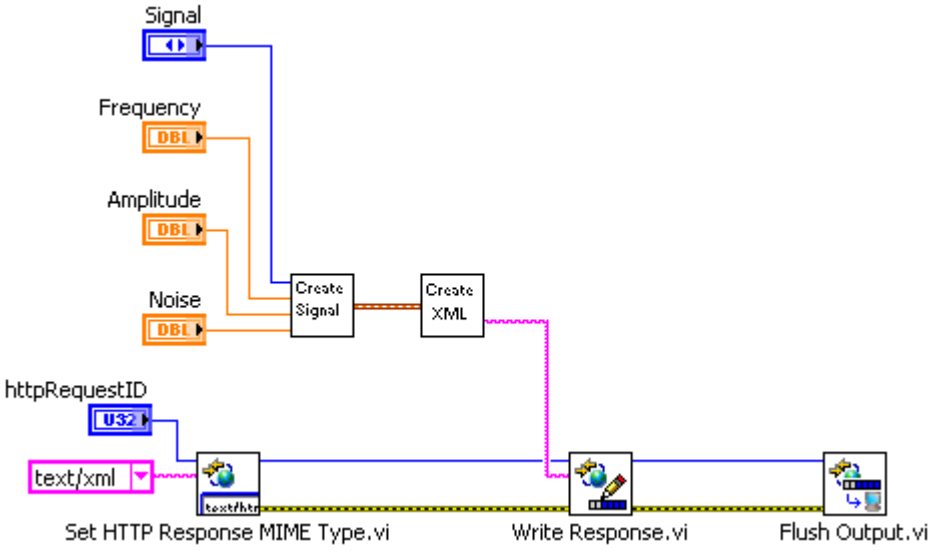

Figura 2.4. Painel Frontal de uma aplicação Labview

2.2.5 Sistemas de Tempo Real (RTE)

Os sistemas de tempo real têm igualmente um papel importante na implementação de laboratórios remotos. Neste subcapítulo far-se-á uma análise de algumas arquitecturas para sistemas de tempo real.

Quando se criam sistemas de tempo de real é necessário que os sistemas sejam robustos o suficiente para serem tolerantes a falhas. O produto final deve ser robusto, fiável e

altamente eficiente. O funcionamento correcto de muitos sistemas de tempo real usados actualmente depende não só dos resultados que estes sistemas produzem, mas também do tempo em que estes resultados são produzidos.

Actualmente o controlo e a monitorização de sistemas de tempo real são feitos usando software e hardware embutido nos próprios sistemas. Esses sistemas de computadores são chamados de sistemas embutidos ("embedded systems"), sistemas de computadores de tempo real ou simplesmente sistemas de tempo real.

Ao contrário dos sistemas de tempo não real, os sistemas de tempo real encontram-se acoplados ao ambiente ou ao sistema a ser monitorizado. Existem muitos exemplos de sistemas de tempo real desde os sistemas de monitorização de sistemas vitais, as novas tecnologias associadas a aviação e a indústria aeroespacial, as redes telefónicas de alta performance, sistemas de realidade virtual, telescópios com sistemas adaptativos entre outros.

Os sistemas de tempo real são uma subdisciplina dos sistemas de computadores, que abrangendo uma grande variedade de áreas diferentes, que vão desde a teoria de controlo, engenharia de software, entre outras áreas. Implementar um sistema de tempo real é necessário ter em atenção vários factores que são os seguintes:

- Selecção do hardware e do software necessário a implementação do hardware
- Compreensão das nuances das linguagens de programação no que diz respeito principalmente as suas capacidades de tempo real.
- Maximizar as capacidades de tolerância a falhas e fiabilidade do sistema.
- Tirar vantagens dos sistemas interoperacionais como os sistemas informáticos baseados no SOA (Service Oriented Architecture) como o CORBA ou os Serviços Web.

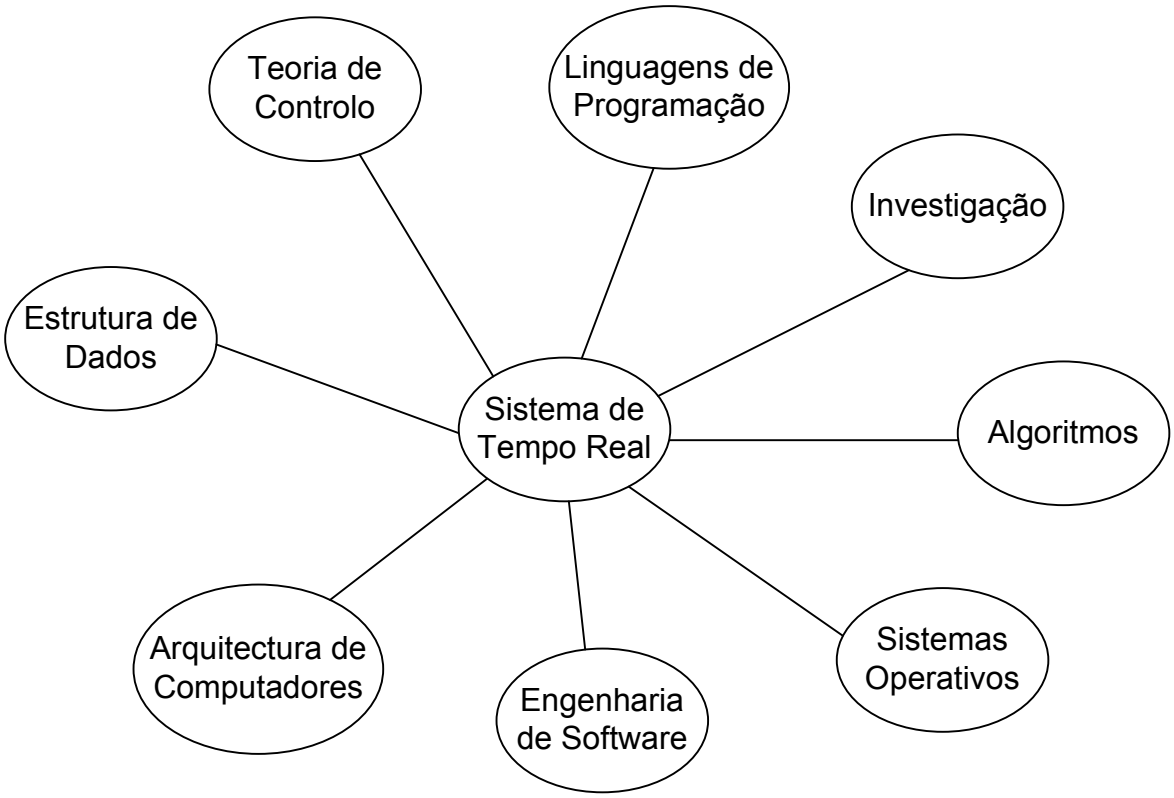

Figura 2.5. Áreas que influenciam os sistemas de Tempo Real

A figura 2.5 (S.Sumathi, P.Sureka, *Labview Based Advanced Instrumentation and Control*) mostra as áreas que alimentam os sistemas de tempo real.

Domínio	Aplicações
Aviónica	Monitores de Navegação
Medicina	Imagem Médica Cirurgia Remota

Sistemas Industriais	Linhas de Montagem Robóticas
Educação	Laboratórios Remotos
Industria Automóvel	Sistemas automóveis

Tabela 2.3. Aplicações Típicas de Sistemas de Tempo Real

A tabela 2.3 mostra algumas aplicações típicas de sistemas de tempo real. Muitas vezes na definição de sistemas de tempo real é importante desfazer algumas definições erróneas. Pode-se sumariza-las seguidamente:

- Os sistemas de tempo real são sinónimos de sistemas rápidos
- Existem metodologias comuns e universalmente aceites para se implementar sistemas de tempo real
- Não existe a necessidade de construir sistemas operativos de tempo real devido a existência de muitos sistemas comerciais.

A primeira definição errónea que diz que um sistema de tempo real tem de ser rápido advém do facto de que muitos sistemas de tempo real trabalham na linha de tempo das décimas de milissegundos, como os sistemas de navegação de um avião. No entanto muitos sistemas de tempo real têm atrasos que podem chegar aos 15 segundos.

A segunda definição atesta que existe uma metodologia universalmente aceite para criar sistemas de tempo real, infelizmente não existe nenhuma arquitectura comum que sirva para implementar qualquer sistema de tempo real. Cada sistema de tempo real é um caso particular, com características muito próprias, dai que a existência de uma

arquitectura comum a todo e qualquer sistema de tempo real, seja na prática muito difícil de se obter.

A terceira definição também não se verifica, muitas vezes os sistemas operativos existentes a nível comercial não servem determinados propósitos, o planeamento e implementação de um sistema de tempo real pode exigir o desenho de um sistema operativo de raiz de modo a acomodar todas as especificidades do sistema a implementar.

2.3 Arquitectura Orientada a Serviços (SOA)

2.3.1 Introdução

A arquitectura orientada a serviços (Service Oriented Architecture - SOA) é um paradigma dos sistemas de informação que permite tirar mais rendimento dos sistemas actuais sem a necessidade de se estar sempre a criar ou inventar novos sistemas. O SOA como arquitectura de integração de sistemas apresenta grandes vantagens, uma vez que a sua adaptabilidade ao tempo e à mudança, são um dos seus pontos fortes, dado que os serviços definidos segundo este paradigma fazem uso, o mais possível de normas, de comunicações como o XML, e permitem às organizações a exposição das suas competências na Internet ou intranets.

2.3.2 Sistemas Informáticos Orientados a Serviços

Os princípios básicos que regem o funcionamento de um sistema baseado no SOA, encontram-se ilustrados na figura 2.7 (Coelho, 2006).

O SOA pode ser descrito da forma seguinte: primeiramente é necessário definir o serviço e fornecer detalhes necessários para quem quiser utiliza-lo o faça de maneira apropriada. Segundo, o provedor de serviços tem de publicar detalhes dos serviços de que dispõe, de modo que os potenciais interessados possam saber como proceder para aceder a esse determinado serviço. Terceiro, os interessados no serviço têm de ter alguma forma para determinar quais os serviços que satisfazem as suas necessidades.

O conceito de serviços permite implementar aplicações sob a forma de conjuntos de serviços que interagem entre si, permitindo ter aplicações em que o acesso a estes

serviços se faz sem a necessidade de permissão para a utilização de um determinado serviço.

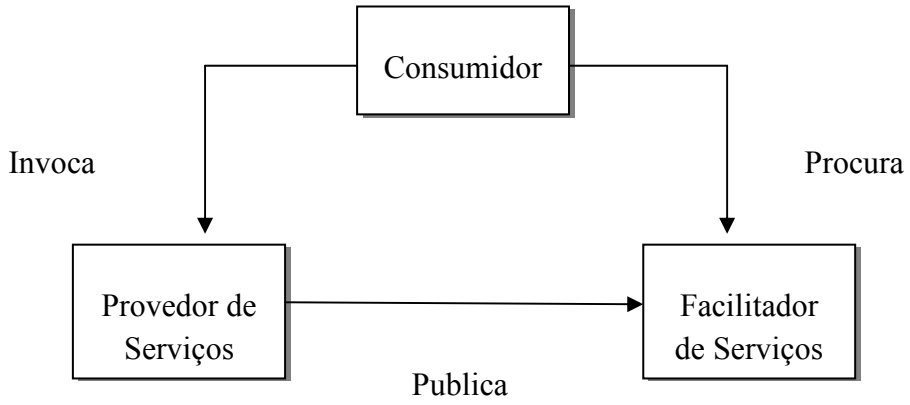

Figura 2.6. Funcionamento do SOA

Um serviço pode ser simples ou composto, isto quer dizer que um serviço composto pode ser definido por um ou mais serviços (camadas de serviços)

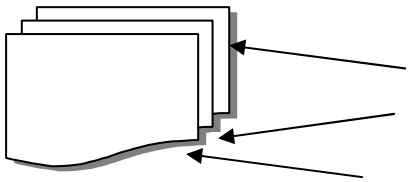

Figura 2.7. Serviços Composto e Simples

Existem várias tecnologias que implementam o SOA, como o p2p ("peer to peer"), GRID Computing, CORBA, DCOM, Web Services etc. A indústria das TI's (Tecnologias de Informação) tem apostado bastante nesta solução, pelo que os Web Services serão a única tecnologia escolhida para a realização do trabalho desta tese.

2.4. Serviços Web (Web Services)

2.4.1 Introdução

Um Serviço Web é uma forma de estrutura cliente/servidor que faz o máximo uso possível dos standards da Internet. Os Serviços Web permitem que sistemas heterogéneos comuniquem entre si por meio de trocas de mensagens.

A inter-operabilidade é garantida fazendo recurso da linguagem XML, que é uma norma da organização W3C (WWW Consortium). Este consórcio tem como objectivo homologar normas que permitam o funcionamento normalizado da Internet.

Uma das grandes vantagens dos Serviços Web, em relação a outras tecnologias para implementação do SOA é que, ao contrário de tecnologias como o CORBA ou a DCOM, os Serviços Web não estão presos a uma arquitectura fechada, dado que ao construir-se uma tecnologia que se baseia em standards bem conhecidos e bastante comuns, o tempo de vida das aplicações aumenta bastante, o que torna mais fácil a integração de sistemas.

2.4.2 XML – eXtensible Markup Language

O XML é uma metalinguagem que serve para definir novas linguagens. O XML não é dependente de nenhuma plataforma e por ser definido em Unicode, permite que seja possível representar conteúdos de muitas linguagens naturais. É devido a estes factores e do facto do XML ser suportado pela maioria dos fabricantes de software que fazem do XML o formato por excelência para a troca de dados entre aplicações heterogéneas; que quer dizer que o XML é o equivalente a linguagem humana para os computadores.

A figura 2.8 mostra um fragmento de código XML

```
<aluno>
    <nome>Marcio</nome>
    <numero>21923</numero>
    <faculdade>UNL</faculdade>
</aluno>
```

Figura 2.8. Fragmento XML

2.4.3 SOAP – Protocolo Simples de Acesso por Objectos

O SOAP é a Framework para mensagens nos Serviços Web. O SOAP permite a integração de aplicações implementadas em tecnologias diferentes. O SOAP apresenta quatro características principais:

- Estrutura de mensagens normalizada baseada no XML;
- Um modelo de processamento que descreve como um serviço processa as suas mensagens;
- Um mecanismo para envio de mensagens SOAP através de diferentes protocolos de transporte na rede;
- Um mecanismo para agregar mensagens não XML em mensagens SOAP;

Estas características ilustram os problemas reais que os Serviços Web têm de resolver, uma mensagem SOAP tem de negociar com diversos protocolos de transporte até chegar ao destino.

O SOAP utiliza um mecanismo de pedido – resposta no qual uma aplicação faz um pedido a outra e recebe uma resposta; tanto o pedido como a resposta são transportados utilizando o formato XML.

O SOAP foi desenvolvido por três empresas Microsoft, Devlopmentor e Userland. A IBM e a Lotus deram continuidade ao trabalho de desenvolvimento do SOAP, tendo obtido a versão 1.1 do SOAP [SOAP 1.1]. Está versão foi proposta e aceite pelo W3C tornando-se assim no standard SOAP actual.

```
<?php
require_once(nusoap.php);
$wsdl="CurrencyExangeService.php";
$client = new soapclient($wsdl,'wsdl');
$param=array(
'country1'=>'usa',
'country2'=>'canada'
);
echo  $client->call('getRate',$param);
```

Figura 2.9. Requisição SOAP em PHP

2.4.4 WSDL – Linguagem de Descrição de Serviços Web

O WSDL pode ser entendido como o vocabulário em XML para descrever Web Services, o WSDL permite que o criador do serviço forneça informação vital sobre como usar o serviço criado.

O WSDL é constituído por 2 partes: uma parte abstracta reutilizável e uma parte concreta. A parte abstracta do WSDL descreve o comportamento operacional dos Serviços Web recontando as mensagens que chegam e saem dos serviços. A parte concreta do WDSL permite ao criador do serviço descrever aonde e como aceder à implementação de um serviço.

Um documento WSDL é um documento XML que contem informações muito detalhadas sobre um determinado serviço. O documento XML deve conter as seguintes características:

- Informações sobre todos os métodos e parâmetros;

- Informações sobre todos os tipos de dados e seus valores contidos no documento XML;
- Informação sobre o protocolo especifico a ser usado;
- Informação sobre o endereço que permite a localização exacta do serviço;

```
<?xml version="1.0"?>
<definitions name="CurrencyExchangeService"
                targetNamespace="http://sd/CurrencyExchangeService.wsdl";
                xmlns:tns="http://sd//CurrencyExchangeService.wsdl";
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns="http://schemas.xmlsoap.org/wsdl">
<message name="getRateRequest">
  <part name="country1"   type="xsd:string"/>
  <part name="country2"   type="xsd:string"/>
</message>
```

Figura 2.10. Fragmento de ficheiro WSDL

Na figura 2.10 encontramos um fragmento de ficheiro WSDL que contem informação sobre a localização do serviço a ser utilizado e o nome do serviço "CurrencyExchangeService"; também está definida a função que executa o serviço "getRateRequest"

2.4.4 UDDI – Descrição Universal de Descoberta e Integração

O objectivo do UDDI é o de representar o nome do serviço e de simultaneamente criar um serviço de nomes, similar ao protocolo DNS ("Dynamic Name Service") que é o protocolo que traduz os endereços em caracteres para endereços IP. Uma especificação UDDI é composta por vários documentos, aonde estão descritos a especificação SOAP API que permite a descoberta e a publicação de operações.

Se a localização de um documento WSDL que descreve um serviço é conhecida, pode-se simplesmente incluir está localização no software de desenvolvimento e implementar a integração deste serviço.

O UDDI permite a criação de um registo de Serviços Web o que permite que dois sistemas possam trocar informação entre si utilizando a Internet. Na figura 2.11 esta representado um exemplo de um arquivo o WSDL.

```
<?xml version=''1.0'' encoding="UTF-8"?>
<definitions name="WeatherService"tions name="WeatherService"
  targetNamespace="http://www.townweather.com/wsdl/WeatherService.ws
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://www.townweather.com/wsdl/WeatherService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <message name="getWeatherRequest">
    <part name="town" type="xsd:string"/>
  </message>
  <message name="getWeatherResponse">
    <part name="temperature" type="xsd:int"/>
  </message>
  <portType name="WeatherPortType">
    <operation name="getWeather">
      <input message="tns:getWeatherRequest"/>
      <output message="tns:getWeatherResponse"/>
    </operation>
  </portType>
  <binding name="WeatherBinding" type="tns:WeatherPortType">
    <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getWeather">
      <soap:operation soapAction=""/>
      <input>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap.encoding/"
          namespace="urn:examples:weatherservice" use="encoded"/>
      </input>
      <output>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap.encoding/"
          namespace="urn:examples:weatherservice" use="encoded"/>
      </output>
    </operation>
  </binding>
  <service name="WeatherService">
    <documentation>WSDL File for Weather Service</documentation>
    <port binding="tns:WeatherBinding" name="WeatherPort">
      <soap:address
        location="http://localhost:8080/soap.servlet.rpcrouter"/>
    </port>
  </service>
</definitions>
```

Figura 2.11. Exemplo de Arquivo WSDL

A forma como o registo de Serviços Web no UDDI esta organizada da seguinte forma:

- **Entidade de Negócio:** Uma entidade de negócios representa informação sobre um determinado sistema. Cada entidade negócio possui uma identificação única, o nome do sistema e uma breve descrição do sistema;
- **Serviço de Negócio:** A entidade de negócio está associada a uma lista de serviços de negócios. Cada entrada desta lista contem uma descrição do serviço, uma lista de categorias que descrevem o serviço e uma lista de apontadores para referências e informações relacionadas;
- **Apontadores de Especificação:** Associado a cada serviço existe uma lista de apontadores que apontam para especificações, e para outras informações técnicas sobre o serviço. Podemos ter um apontador direccionado para um URL que fornece indicações sobre como invocar o serviço. Estes apontadores também podem aceder ao Service Level Agreements (SLA) que descrevem as condições de acesso ao serviço. Os apontadores de especificação também associam o serviço com um tipo de serviço;
- **Tipos de Serviço: E**specifica informações tais como o nome do serviço, o nome da organização que o publicou, a lista de categorias que descrevem o serviço e apontadores para especificações técnicas do serviço, tais como as definições de interface, formatos de mensagens e protocolos de segurança;

2.4 Composição de Serviços Web

2.5.1 Introdução

As aplicações informáticas baseadas no paradigma de arquitecturas orientadas a serviços requerem cada vez mais a composição de serviços. Com a composição de serviços tanto os utilizadores dos serviços como quem providencia os serviços, podem compor serviços simples ou básicos, de modo a criar serviços mais complexos.

Esta composição pode ser descrita de duas formas: orquestração e coreografia. Orquestração refere-se é um processo interno aonde há interacção com Serviços Web tanto internos como externos. Coreografia controla a sequência de mensagens entre dois sistemas e não no interior dos sistemas.

2.5.1 Requisitos de Composição de Serviços

A composição de serviços oferece ao provedor de serviços a possibilidade de reutilização de serviços e ao consumidor de serviços (Coelho, 2006), oferece um acesso fácil e transparente a serviços complexos. O provedor de serviços, tal como o consumidor terem apenas acesso a descrições funcionais dos serviços providenciadas pela linguagem WSDL. Logo a sua composição tem de obedecer a determinados requisitos já que os serviços são executados em containeres diferentes separados por firewalls e outras barreiras de segurança. Estes requisitos são os seguintes:

- Conectividade
- Propriedades não funcionais de qualidade de serviço (QoS)
- Escalabilidade

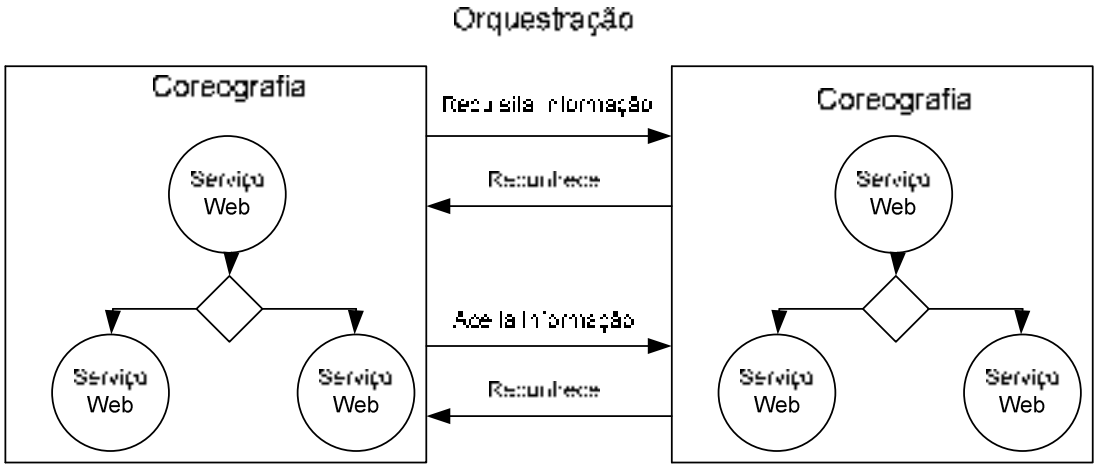

Figura 2.12. Orquestração e Coreografia

É importante realçar que a invocação assíncrona de serviços é vital para garantir a escalabilidade e a segurança que os ambientes de tecnologias de informação actuais requerem. A invocação de serviços concorrenciais permite também o aumento da eficiência dos sistemas de uma forma global.

É também importante garantir que a orquestração de Serviços seja dinâmica e adaptável de modo a suportar mudanças na estrutura do sistema de serviços. Deve também haver uma separação clara entre o funcionamento do processo propriamente dito e os serviços Web. Deve ser também possível a composição de serviços de mais alto nível a partir de processos previamente orquestrados. Pode-se então fazer assim uma combinação recursiva.

A figura 2.2 ilustra a composição de serviços (Coelho, 2006).

2.5.2 BPEL – Business Process Execution Language

O BPEL é uma especificação lançada pelas seguintes empresas: Microsoft, IBM, Siebel, BEA e SAP em Maio de 2003. A especificação BPEL modela o comportamento de Serviços Web numa interacção de negócios.

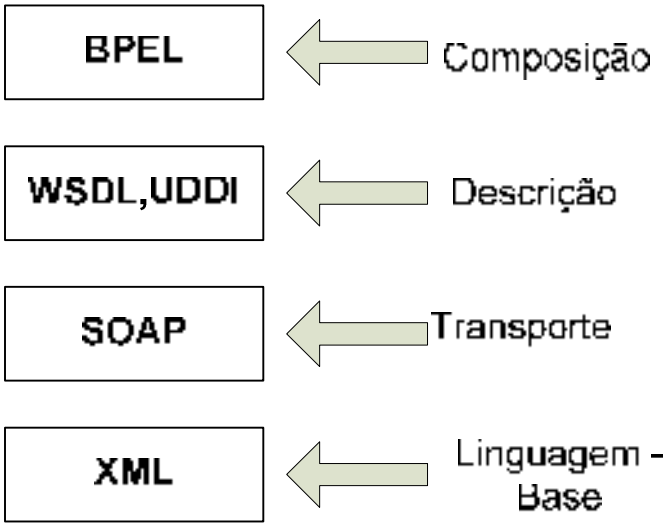

Figura 2.13. Camadas Componentes de Serviços Web

Na figura 2.13 (Coelho, 2006) estão representadas as camadas que compõem os Serviços Web no paradigma do SOAP. Nesta figura notamos que a camada BPEL é uma camada acima da camada WSDL. A camada WSDL define as operações permitidas no modelo de negócio e a camada BPEL permite sequenciar essas operações. Por outro lado, a camada WSDL pode referenciar serviços externos necessários à camada BPEL.

A especificação BPEL suporta acções básicas e complexas. Um exemplo de acção básica pode ser a responder a pedidos de serviços. A figura 2.7 explicita de uma

forma clara a maneira como funcionam os Serviços Web segundo o paradigma SOAP, percebe-se assim a estrutura completa deste sistema e os níveis de funcionamento.

2.5 REST – Representação por Transferência de Estado

2.6.1 Introdução

O REST é um estilo de arquitectura de software alternativo ao SOAP para implementar Serviços Web. Um dos motivos pelo qual a arquitectura REST está a aumentar de popularidade é a sua simplicidade e facilidade de uso bem como o uso extensivo de tecnologias Web nativas como o HTTP. O REST ao contrário do SOAP não é um standard e muito menos é reconhecido pelo W3C. Os princípios chave do REST são os seguintes:

- O conceito de recursos (ex: um documento é um recurso);
- Cada recurso tem um único ID (ex: o url do documento);
- Os recursos podem estar relacionados;
- Uso de standards (ex: HTML, HTTP, XML);
- Um recurso pode ter múltiplas formas;
- A comunicação é feita sem estados usando o HTTP;

Os princípios do REST foram desenvolvidos por Roy Fielding (investigador ligado ao HTTP) na sua tese de Doutoramento (Fielding, 2000).

Em alguma literatura existe a tendência para não considerar o REST um serviço Web. Contudo é importante reparar que o REST é uma arquitectura orientada a serviços (SOA) apenas não obedece ao paradigma do SOAP. Logo no REST não existem as camadas SOAP, UDDI, WSDL e BPEL. O REST utiliza o protocolo HTTP e as primitivas do HTTP para aceder a recursos. A grande diferença entre o REST e o SOAP é que no REST temos um maior controlo dos dados que recebemos. Ao efectuarmos uma requisição usando o verbo ou primitiva GET do protocolo HTTP, estamos a fazer o pedido directo de um determinado dado concreto. No SOAP o uso dessa mesma

primitiva GET não se traduz numa transferência de dados concretos. O SOAP depende de uma API ou seja depende de uma semântica complexa que tem de ser cumprida rigorosamente, no REST tem-se acesso directo aos dados ou recursos que se pretende usar sem a necessidade de se cumprir um conjunto de regras ou um contracto definido num ficheiro WSDL. No REST tudo o que se precisa para se consumir um serviço ou recurso é o seu URL. Tendo o URL tem-se imediatamente acesso aos serviços ou serviço disponível. Este tipo de abordagem do REST torna mais simples o acesso aos serviços criados segundo este paradigma.

Podemos resumir as diferenças entre os dois paradigmas da seguinte forma (Samisa Abeysinghe, *RESTful PHP Web Services*):

- O REST é fundamentalmente gestão de dados, já o SOAP é gestão da API;
- O REST tem um conjunto de semântica predefinida que usa os verbos HTTP; (GET, PUT,DELETE, POST), URL's, recursos e representações;
- A semântica do SOAP está definida no ficheiro WSDL;
- O REST manipula, de forma dinâmica os URL's criados para os recursos;
- URL's no paradigma REST representam uma referência a um recurso;

2.6.2 Arquitectura Orientada a Recursos (ROA)

O REST introduz o conceito de recurso ou arquitectura orientada a serviços. Os servidores possuem recursos e os clientes consomem recursos, um recurso pode ser qualquer informação a qual se possa atribuir um nome. Pode ser um documento, uma imagem, um vídeo. O URL é usado como identificador do recurso ou seja o URL é o nome do recurso. O protocolo HTTP é utilizado pelo REST para aceder a recursos de todos os tipos. Chama-se serviço orientado ao recurso a um serviço construído de acordo com o estilo de desenho REST. O REST na sua essência manipula o protocolo

HTTP para alterar o estádo do recurso. O REST não requer o uso de uma API como o SOAP. Uma das principais vantagens do REST é que o cliente e o servidor estão totalmente desacoplados. Isso quer dizer que a implementação do cliente é totalmente independente. A vantagem deste paradigma REST é que quando se junta o cliente e o servidor o sistema funciona sem haver necessidade de uma grande quantidade de testes.

É importante perceber-se que o REST não é uma arquitectura mas sim um estilo de desenvolver software que permite desenvolver Serviços Web obedecendo a determinados critérios.

Recurso no paradigma do ROA é definido como algo ou qualquer coisa que é importante o suficiente para ser referida individualmente. Um recurso tem um endereço associado a ele. Um recurso pode ser algo que pode estar armazenado num computador, um documento, um campo de uma base de dados, o resultado da execução de um algoritmo. A questão importante que permite que um recurso, seja de facto um recurso no paradigma REST é o facto de este recurso ter associado pelo menos um endereço URL. Portanto a condição necessária para um recurso estar disponível na Internet é a sua presença na Internet.

O paradigma ROA tem 4 características base (Samisa Abeysinghe, *RESTful PHP Web Services*):

- **Endereçabilidade:** Uma aplicação é endereçável se expõe os seus dados como recursos. Para o utilizador final o endereçamento é a face visível do recurso, a endereçabilidade deficiente torna difícil senão impossível o acesso ao recurso;
- **Ausência de estados:** ausência de estado ou statelessness (“inglês”) significa que cada pedido HTTP, está isolado ou seja não depende do pedido seguinte ou do que o precede. No pedido que o cliente faz está incluída toda a informação necessária para que o servidor satisfaça o pedido. A ausência de estados elimina a necessidade de haver controlo de falhas o que garante a simplicidade do sistema. O protocolo HTTP é um exemplo de um sistema em que não há

estádos. A vantagem da ausência de estádos é que, em termos de escalabilidade, podemos ter recursos alojados em servidores diferentes caso seja necessário;

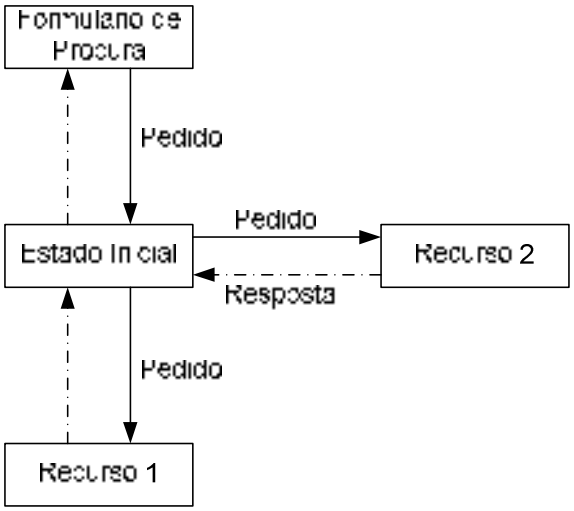

Figura 2.14. Sistema de busca sem Estádos

Na figura 2.14 (Leonard Richardson and & Sam Ruby, *Restful Web Services*) temos um sistema de procura sem estádos. Este sistema pode representar um motor de busca no qual o cliente faz um pedido para aceder a recursos. Os pedidos para aceder ao recurso 1 ou ao recurso 2 são independentes.

Na figura 2.15 (Leonard Richardson and & Sam Ruby, *Restful Web Services*) está representado um sistema de procura com estádos neste caso a procura dos recursos não é independente, os dois recursos têm de estar alojados no mesmo servidor

- **Conectividade:** Define-se como um sistema com conectividade a um sistema que possua hiperligações. Ou seja podemos ter um Serviço Web que é uma colecção links

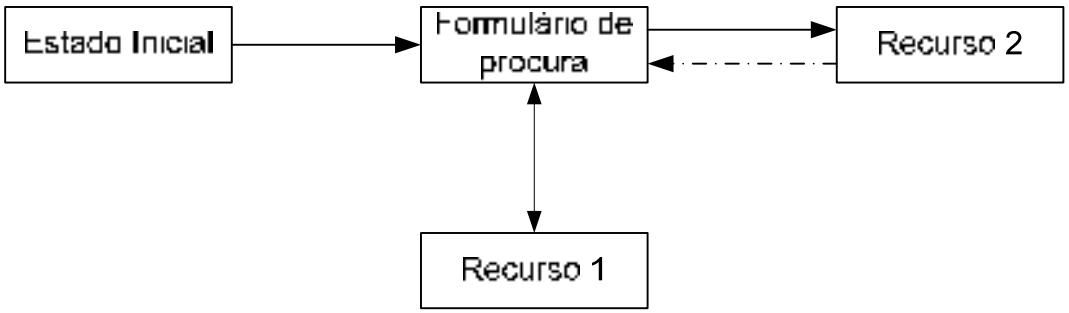

Figura 2.15. Sistema de Busca com Estádos

que podem acedidos como se fossem estádos. O que se pretende é que cada recurso esteja ligado a outros recursos por meio de hiperligações.

- **Interface Uniforme:** O protocolo HTTP e as suas primitivas, GET, POST, PUT e DELETE fornecem uma interface ao REST, mas o que torna essa interface uniforme é o facto de todos os serviços usarem a interface da mesma forma. Sem uma interface uniforme que nos permita por exemplo usar a primitiva GET, que na linguagem Web significa um READ, e significa sempre isso, qualquer que seja o recurso que está a ser utilizado. Sem a interface uniforme seria sempre que se acedesse a um recurso seria necessário descobrir como cada serviço recebe e envia informação. Com a interface esse problema esta resolvido, quaisquer dois serviços enviam e recebem informação sempre da mesma maneira.

2.6.3 Serviços Web e o protocolo HTTP

O protocolo HTTP é um protocolo de transporte. Geralmente o desenvolvimento Web utiliza o HTTP como meio de transporte e o XML como formato de mensagem. O XML não é o único formato de mensagem disponível na Internet, existem outros como o JSON, RSS, Atom, CSV. No entanto o XML é o formato mais popular e por isso o mais usado. Quando um utilizador utiliza o browser está de facto a aceder a recursos usando o URL, o browser faz o uso da primitiva GET do protocolo HTTP para aceder ao

recurso no servidor, o servidor responde com uma mensagem com o conteúdo da resposta que pode ser em qualquer um dos formatos anteriormente indicados. O REST é um serviço Web que funciona por cima do HTTP, ao acedermos a um recurso via URL e ao obtermos a resposta em formato XML estamos de facto a fazer uso dos princípios do REST. O Labview na sua versão 8.6 usa precisamente esse princípio para oferecer recursos que possam ser consumidos e processados por programas terceiros.

2.6.4 Métodos HTTP

O HTTP é o protocolo de transporte por excelência da Internet, muito devido a sua simplicidade e ao facto de ser usado em praticamente todas as plataformas. O HTTP pode ser usado para aceder a vários tipos de recursos como vídeo, imagem, páginas HTML e aplicações. Quando se acede a algum recurso via HTTP é enviado o identificador do recurso assim como a acção a ser desempenhada nesse recurso. O URL identifica o recurso e a acção é definida usando primitivas ou verbos HTTP. O HTTP apresenta um conjunto de verbos cuja semântica ajuda a identificar o tipo de acção esperada no recurso.

Verbo	**Descrição**
GET	Recupera um recurso dado o seu URL
POST	Envia um recurso para o servidor. Actualiza o recurso na localização dada pelo URL
PUT	Envia um recurso para o servidor a ser guardado na localização dada pelo

	URL
DELETE	Apaga um recurso dado o seu URL
HEAD	Recupera os meta dados de um recurso dado o seu URL

Tabela 2.4. Primitivas HTTP

Através deste verbo HTTP pode-se entender de que forma podemos interagir com os recursos disponibilizados. Os verbos do HTTP permitem criar uma interface uniforme para interagir com os recursos o que vai de encontro aos princípios da arquitectura de estilo REST. No contexto REST o POST deve ser usado para fazer actualizações de recursos. O GET, o HEAD e o POST são usados para se consumir os recursos durante o seu ciclo de vida e o DELETE termina o ciclo de vida do recurso. No entanto na prática o GET é largamente o verbo mais utilizado, isto devido a constrangimentos na utilização dos outros verbos, uma vez que por exemplo o POST e o DELETE alteram recursos e logo poderiam ser usados de forma incorrecta, no sentido em que poder-se-ia por acidente apagar um recurso, acção que é grave. Dai que alguns cuidados a esse nível devem ser tomados.

2.6.5 Exemplo de Aplicação

A versão 8.6 do Labview permite criar Serviços Web que obedecem ao paradigma REST, um exemplo básico passa pela criação de um exemplo básico cuja figura apresenta-se a seguir:

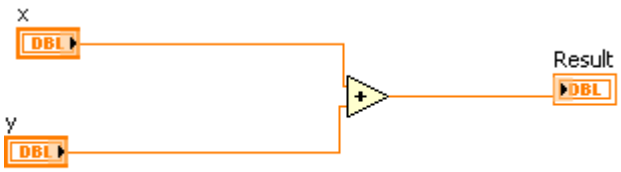

Figura 2.16. vi de um somador criada em Labview

Neste exemplo 2 parâmetros x e y são colocados a entrada de um somador, o resultado da soma é guardado na variável Result, está vi (nome que se da aos esquemas criados em Labview) é o ficheiro fonte do recurso REST que se cria com a ajuda do Labview. O resultado desta operação será um serviço com uma identificação única, neste caso o URL gerado pelo Web Service consequente. Este Web Service gerará um ficheiro XML que permitirá que qualquer cliente que funcione de acordo com o paradigma do REST possa consumir esse recurso. A resposta pode ser no formato XML, TXT ou HTML, o XML é o escolhido por ser mais comum na Internet e também por ser aquele que é mais fácil manipular por aplicações clientes, nomeadamente extracção de dados contidos nos nós do ficheiro XML.

```
<Response>
<Terminal>
<Name>Result</Name>
<Value>3,0000</Value>
</Terminal>
</Responde>
```

Figura 2.17. Resposta do servidor no formato XML criado pelo Labview

Na figura 2.19 o código de um client orientado aos recursos (REST) em linguagem PHP, que irá consumir o serviço préviamente criado pelo Labview, o serviço recebe dois parâmetros, neste caso dois valores numéricos e devolve o resultado da sua soma

ao cliente, o resultado é imprimido num ficheiro XML que é depois tratado pela função file_get_contents.

```
<?php
$x=1;
$y=2;
$url='http://localhost/soma/Add/'.'/'.$x.'/'.$y;
$xml=file_get_contents($url);
echo $xml;
?>
```

Figura 2.18. Cliente REST em PHP

2.6 Exemplos de Arquitecturas para Laboratórios Remotos

2.7.1 Introdução

A necessidade de se fazer uso da Internet para se aceder a recursos remotamente não é um tema novo. Existem vários exemplos de laboratórios remotos tanto em Portugal como nos mais diversos países do mundo. Neste subcapítulo vai fazer-se uma apreciação de alguns sistemas de acesso remoto construídos e disponíveis em reputadas universidades europeias e mundiais e das tecnologias envolvidas na implementação destes sistemas. Os sistemas a analisar são o WebLab Deusto da Universidade de Deusto, o iLab do Massachussets Institute of Technology e o Remote Lab da Faculdade de Engenharia da Universidade do Porto.

2.7.2 Deusto WebLab

O WebLab Deusto (HTTPs://www.weblab.deusto.es) é um laboratório remoto desenvolvido por equipa de investigadores da universidade de Deusto em Espanha. O WebLab Deusto é um projecto que se iniciou no ano de 2001 e inicialmente estáva focado na área da electrónica programável sendo que actualmente está a ser usada em 4 áreas e suporta experiências baseadas em CPLD, FPGA, controladores PIC e instrumentação GPIB servindo centenas de estudantes. O WebLab Deusto inclui funcionalidades como acesso vídeo para que os utilizadores possam observar a experiência, envio e recepção de ficheiros entre o cliente e o servidor. O WebLab Deusto utiliza Phyton para o desenvolvimento da aplicação servidor. Para a aplicação cliente o WebLab Deusto utiliza o AJAX. Para aplicação cliente o ambiente de desenvolvimento é o open-laszlo que permite o desenvolvimento de código numa única

linguagem sendo que a saída pode ter vários runtimes em flash (várias versões), AJAX e J2ME.

A figura 2.20 (Javier Garcia – Zubia, Pablo Orduña, Diego López-de-Ipiña, Unai Hernandez, Iván Trueba, *Remote Laboratotories from the Software Engineering Point of View*, University of Deusto) está ilustrada a arquitectura do WebLab Deusto; nesta arquitectura os servidores 1 e 2 estão ligados aos processos laboratoriais propriamente ditos e também a Webcams. Uma das vantagens do WebLab Deusto em usar o openlaszlo é precisamente a possibilidade de programando apenas numa linguagem a linguegem LZX, o software produz facilmente runtimes em 3 tecnologias diferentes (flash, J2me e ajax), tornando assim mais abrangente o laboratório remoto.

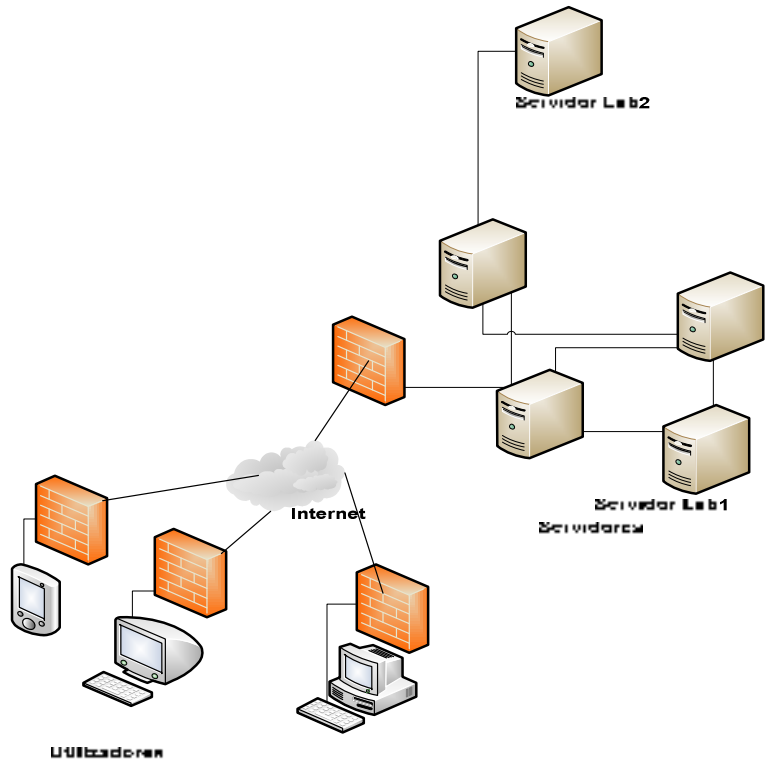

Figura 2.19. Arquitectura WebLab Deusto

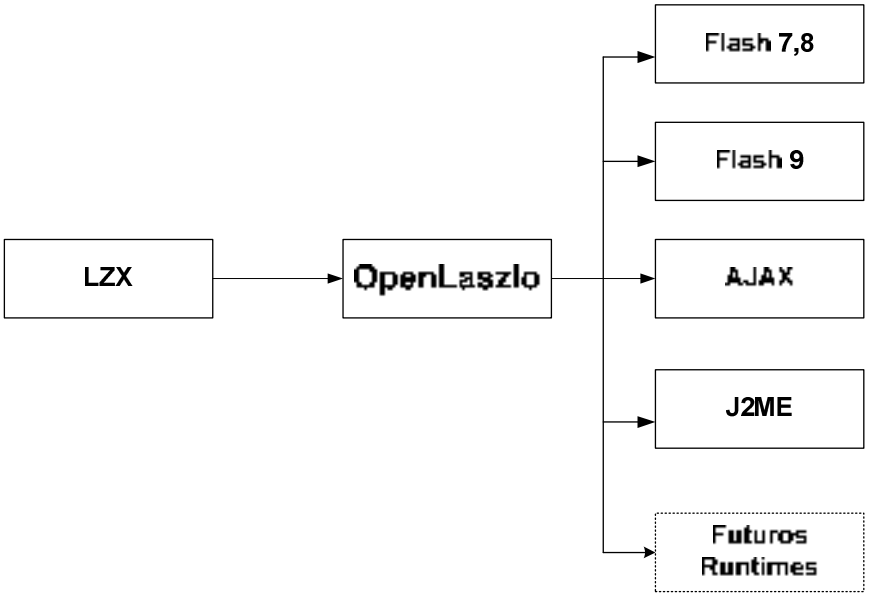

Figura 2.20. Plataforma Openlaszlo

Na figura 2.20 está representada o esquema da plataforma openlaszlo usada para programar os diversos clientes do WebLab Deusto.

2.7.3 iLab

O projecto iLab (HTTP://ilab.mit.edu) foi desenvolvido no MIT, por cientistas do MIT e da Microsoft e faz parte do projecto iCampus. O objectivo do iLab é o de integrar os laboratórios acessíveis pela Internet na educação dos estudantes do MIT e de universidades parceiras, assim como de parceiros industriais do MIT. O iLab foi assim projectado para cumprir os seguintes objectivos:

1. Criar uma infra-estrutura em software que permita que de uma forma fácil e barata, um laboratório remoto possa estar acessível;
2. Providenciar um ambiente que possa providenciar múltiplos laboratórios remotos de forma continua;
3. Tornar viável a partilha de recursos físicos, isto significa que duas universidades poderão partilhar o mesmo laboratório físico ao qual poderão

aceder de forma remota em simultâneo o que reduz os custos para ambas as partes;

A figura 2.22 (HTTP://ilab.mit.edu) apresenta a arquitectura do iLab. Nesta arquitectura cada laboratório apresenta o seu próprio servidor, que está ligado ao service broker via Serviços Web.

Cada laboratório possui o seu servidor independente e está ligado ao service broker (registo de serviços) via Serviços Web, é nesse servidor que passa toda a gestão da ligação ao hardware do laboratório. O service broker medeia a relação entre o servidor do laboratório e o cliente para além de providenciar serviços de armazenamento e administração a todos os laboratórios do campus, para além disso fornece aos clientes as interfaces para acesso aos recursos dos diversos laboratórios remotos presentes no sistema iLab

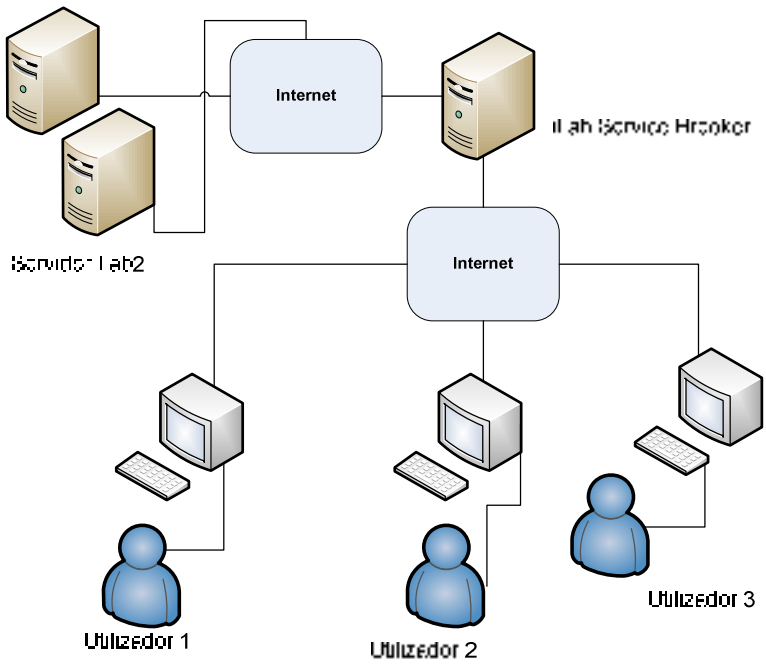

Figura 2.21. Arquitectura do sistema iLAB – MIT

2.7.4 remotelab

O remotelab (HTTP://remotelab.fe.up.pt) é um laboratório virtual desenvolvido pela Faculdade de Engenharia da Universidade do Porto. O remotelab apresenta diversas experiências desenvolvidas em Labview pelo que o download do pluggin do Labview é requerido visto que foram desenvolvidas nesse software. A arquitectura do remote lab está representada na figura 2.23.

O remotelab faz uso das capacidades que o software Labview tem, nomeadamente do facto de já possuir um servidor Web que permite publicar as experiências via Internet.

Esta arquitectura apresenta o problema do facto de que necessita da parte do cliente do download do pluggin da Labview com certa de 70MB, o que de certa forma reduz o desempenho do sistema e limita o acesso de utilizadores principalmente utilizadores que não possuam privilégios para alterarem ficheiros de sistema nas suas máquinas.

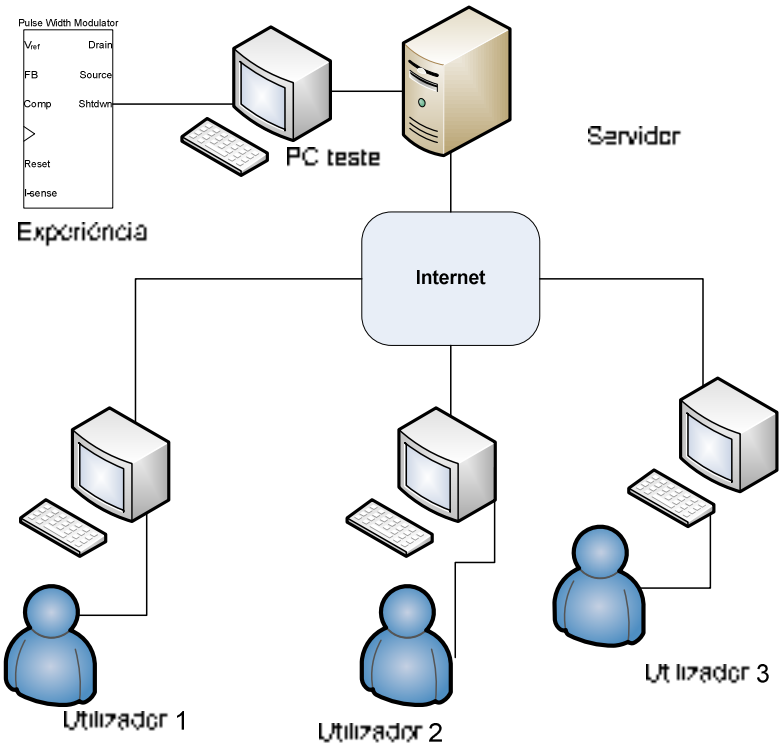

Figura 2.22. Arquitectura do remotelab

2.8 Conclusões

Neste capítulo vimos as tecnologias que são relevantes para a realização desta dissertação, de uma maneira geral a arquitectura baseada em Serviços Web afigura-se como a mais promissora na parte do servidor, dado que resolve muitos problemas relacionados com questões de integração de sistemas diferentes. Na parte do cliente a utilização do AJAX apresenta-se como a mais promissora tendo a conta as vantagens que traz para o sistema globalmente, tanto em termos de poupança de largura de banda como também na eficiência para o browser do utilizador. No próximo capítulo é apresentada a arquitectura a partir da qual se implementará o laboratório remoto.

3 Arquitectura Proposta baseada em Restful Services

3.1 Introdução

Neste capítulo irá ser referido em pormenor a arquitectura escolhida para a realização de um laboratório remoto. A arquitectura será implementada em Serviços Web utilizando o estilo de arquitectura de software RESTful em Labview na parte do servidor e usando ajax na parte do cliente. Os recursos utilizados ao nível de software foram tanto quanto possível com recurso a soluções open – source, no entanto a utilização do software Labview versão 8.6 foi necessária para o desenvolvimento dos serviços e da aplicação que controla os processos em estudo, no caso um filtro de segunda ordem.

3.2 Arquitectura Proposta

3.2.1 Introdução

A arquitectura proposta pode ser descrita de forma resumida através da figura 3.1, na qual estão presentes três elementos que definem o funcionamento do sistema. Temos a aplicação Labview que está fisicamente ligada à experiência; esta aplicação é o ficheiro fonte que vai permitir a criação do Serviço Web, que por sua vez executa os pedidos vindos da interface Web. O Serviço Web utiliza os protocolos e normas da Internet, o que permite uma interacção imediata com o cliente (browser). Com esta arquitectura obtemos um sistema que é globalmente leve, uma vez que o cliente necessita apenas do

browser sem qualquer tipo de tecnologia adicional; tal melhora a eficiência do sistema uma vez que do lado do cliente temos uma aplicação cliente leve.

Figura 3.1. Arquitectura Resumida

A aplicação cliente é integralmente programada utilizando AJAX, que pelas suas características próprias permite que o utilizador final utilize apenas aquilo que já tem acesso para se ligar à Internet, o browser. Com está solução consegue-se tirar partido dos meios já existentes o que torna mais apetecível e mais fácil a utilização de um sistema de laboratório remoto que se baseie nesta solução.

3.2.2 **Arquitectura do Sistema Implementado**

Para melhor explicar a implementação da arquitectura considere-se o seguinte esquema, que explicita o funcionamento global do sistema implementado.

Atendendo a arquitectura representada na figura 3.2 pode-se dizer que o protocolo base de comunicação usado na mesma é o protocolo HTTP. É importante salientar que as componentes Interface AJAX e Labview Rest Service são aplicações "server-side" correm do lado do servidor. O utilizador envia pedidos de dados a interface AJAX que do seu ponto de vista é uma página HTML eventualmente com elementos dinâmicos (actualização de dados dinâmica). A interface AJAX envia esse pedido ao servidor utilizando o objecto XMLHTTPRequest que é a parte da tecnologia AJAX que permite a troca de dados assíncrona com o servidor.

A resposta que vem do servidor poder ser tanto em formato XML, TXT ou HTML. Apesar do objecto XMLHTTPRequest ter o XML no nome, o XMLHTTPRequest está preparado para suportar e tratar dados recebidos em qualquer um dos formatos mencionados acima

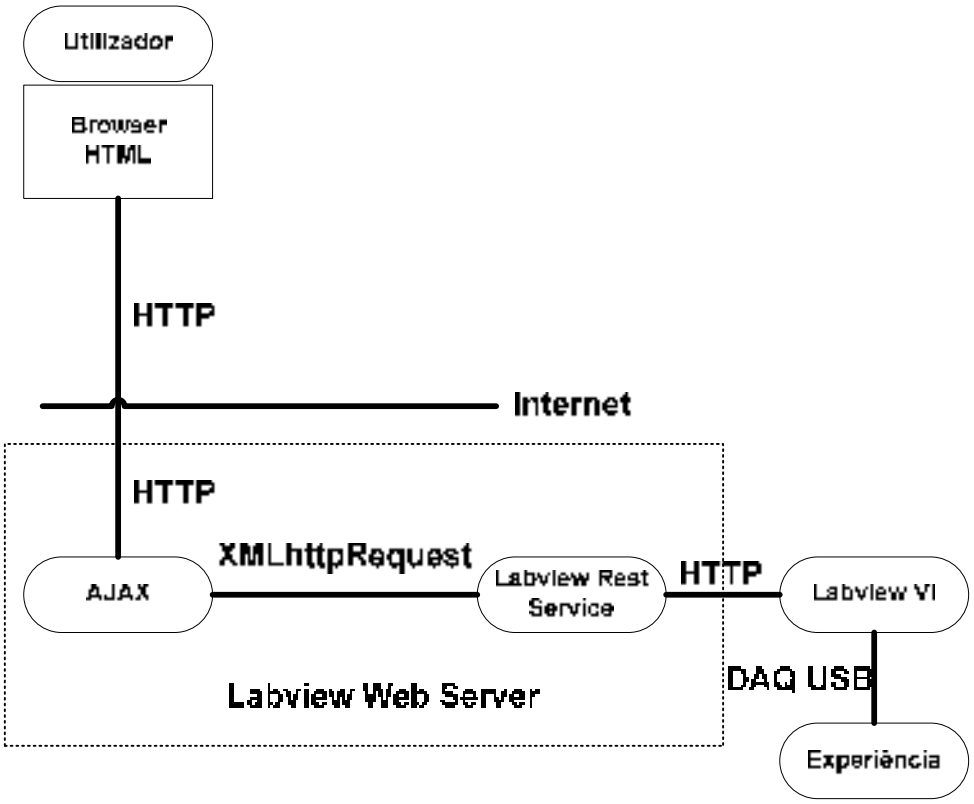

Figura 3.2. Arquitectura do Sistema Implementado

. A comunicação entre o servidor e o Serviço Web (REST Service) faz-se igualmente utilizando o protocolo HTTP. A comunicação entre o Serviço Web e a aplicação de controlo desenvolvida no Labview faz-se utilizando "Web Methods VI". Como vimos anteriormente dá-se o nome de VI's às aplicações criadas na linguagem de programação visual do Labview, as "Web Methods VI's" são VI's especiais que partilham variáveis com as VI's que são usadas para efectuar o controlo dos experiências, essas VI's permitem que as variáveis de controlo e os indicadores usados nas VI's estejam acessíveis pela Internet usando o protocolo HTTP. Quando no Labview configura-se um Serviço Web, são as "Web Methods VI's" que servem de ficheiro fonte para a

configuração do Serviço Web, na figura 3.5 temos o diagrama da “Web Method VI” que foi utilizada para exportar a VI principal como serviço.

A aquisição de dados vindos do experimento é feita com recurso a uma placa de aquisição de dados com interface USB (placa USB-6009 da National Instruments).

4. Resultados Experimentais

Importa agora fazer uma descrição mais detalhada dos elementos que compõem a arquitectura, nomeadamente da interface AJAX, da aplicação de controlo (Labview VI), da experiência e do Serviço Web (também denominado Serviço REST). No geral pode-se afirmar que se trata de uma arquitectura eficiente, na medida em que garante o uso de poucos recursos e é de baixa complexidade, cumprindo contudo o seu objectivo que é o de estar de acordo com o paradigma dos Serviços Web neste caso baseados no REST. Fornece também serviços que possam ser acedidos de forma normalizada, ou seja utilizando os protocolos já conhecidos dos clientes (browsers), nomeadamente o HTTP.

4.1.1 Aplicação de Controlo

A aplicação de controlo é a aplicação projectada em linguagem visual que faz a ligação entre o hardware de teste que neste caso é um sistema (filtro) de primeira ordem e o Labview. Está aplicação ou VI (nome dado as aplicações feitas em Labview) será a responsável por todo processo que tenha com a aquisição e o envio de sinais para o sistema de 1ª ordem. O processo de aquisição de sinais faz-se usando uma placa de aquisição de dados com ligação USB ao servidor.

O Labview, por ser um software que segue o paradigma da instrumentação virtual, dispõe de vários instrumentos virtuais que permitem a visualização de sinais. Recorrendo a um desses instrumentos virtuais que permitem a visualização de sinais temos acesso a resposta do filtro de 1ª ordem de forma imediata. Para além disso podemos aceder ao sinal de saída mesmo não usando os instrumentos de visualização, bastando para isso utilizar variáveis que possam receber o sinal de saída e mostrá-lo numericamente. A utilização de variáveis mais do que fazer o gráfico do sinal de saída

do filtro de 1ª ordem é que vai permitir o acesso exterior ao sistema. Na figura 4.1 está representado o diagrama de blocos da VI principal.

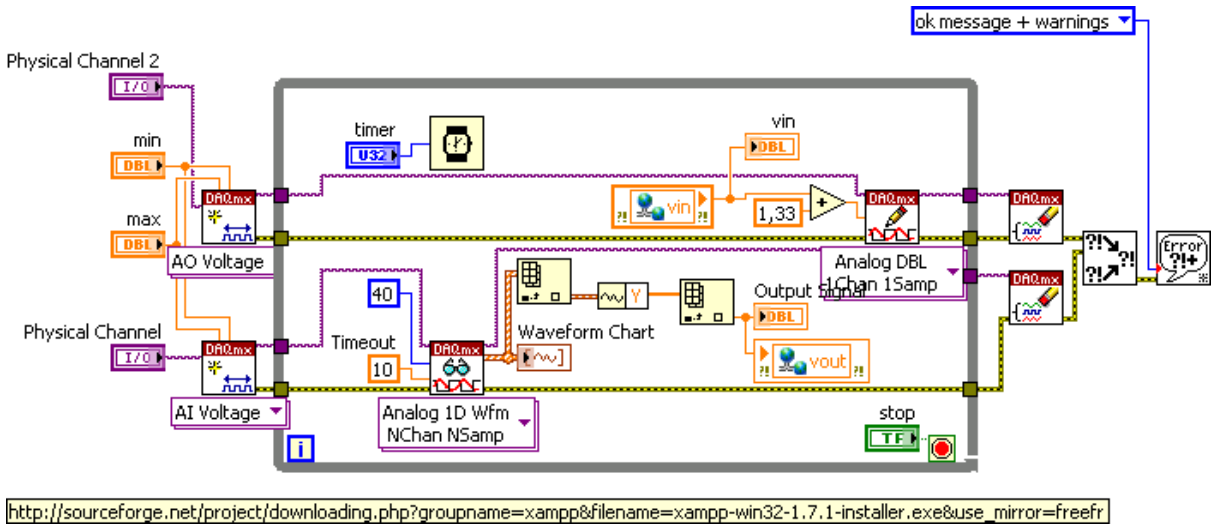

Figura 4.1. Diagrama de Blocos da aplicação principal (filtro.vi)

4.1.2 Funcionamento da aplicação

O funcionamento da aplicação é bastante simples, e está dividida em 3 partes: aquisição de sinal, processamento e apresentação. A aquisição do sinal é feita com recurso a dois blocos de aquisição, um para o canal de entrada e outro para o canal de saída. É importante realçar que as aplicações em Labview têm de correr no interior de uma estrutura, neste caso usou-se um "while" para garantir que a aplicação possa funcionar de modo contínuo.

Depois da aquisição do sinal é necessário ligar um bloco de escrita ao canal de saída da placa que está ligado ao canal de entrada do filtro de 1ª ordem. É neste canal que será injectado o sinal de entrada "vin" que é um sinal de tensão que varia entre 0V e 5V (que são os limites máximos da placa de aquisição do tipo USB 6009 da National Instruments). Ao canal de entrada temos de ter um bloco de leitura que para podermos ler o sinal que vem da saída do circuito de 1ª ordem. Á leitura deste sinal faz-se ligando

um bloco gráfico que leia sinais ou ondas. A saída do bloco de leitura para além da ligação de um bloco gráfico, ligou-se também um bloco chamado de "índex array"; a sua função é a de fazer a separação do sinal multidimensional nas suas componentes unidimensionais (indexar). Depois de indexarmos os sinais é necessário extrairmos a componente que pretendemos; neste caso é a componente y, e depois de separarmos o sinal y temos de fazer a conversão "cast" para o tipo de dados double e escrever na variável de saída. Neste diagrama temos duas variáveis "vin" e "vout" de um tipo especial cujo ícone é o seguinte:

Figura 4.2. Ícone de variável partilhada

Estás variáveis são as chamadas variáveis partilhadas "shared variables". O Labview permite uma funcionalidade muito importante que é a criação de variáveis partilhadas "shared variables" (na figura 4.2 está representada uma variável partilhada); estas variáveis podem ser usadas por mais do que uma VI, elas são particularmente importantes quando pretendemos aceder a estás variáveis externamente utilizando Serviços Web. Para que estás variáveis estejam acessíveis a aplicações externas ao Labview é necessário criar uma outra VI que tem de estar no mesmo projecto da VI principal. Esta VI partilha as mesmas variáveis de controlo com a VI principal. Desta forma podemos controlar a VI principal apenas alterando as suas variáveis por intermédio da VI auxiliar ou "Web Method VI".

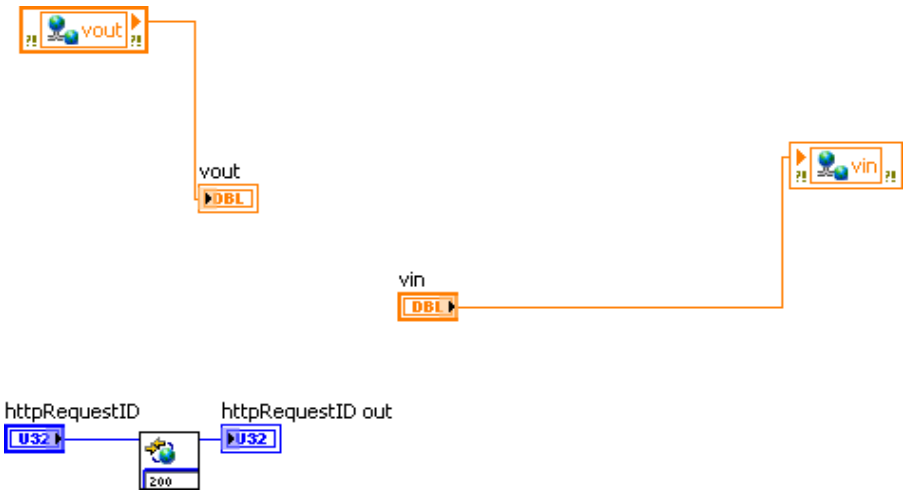

Figura 4.3. Diagramas de blocos da VI auxiliar ou Web Method VI.

Como se pode analisar na figura 4.3, o diagrama de blocos da VI auxiliar é muito simples possuindo apenas as duas variáveis separadas "vin" e "vout" e um bloco que faz o controlo das ligações HTTP, que é um bloco que faz parte da nova palete do Labview para Serviços Web. Este bloco é do tipo "Web Service VI" a sua função como o seu nome indica "set HTTP response code" é atribuir um código que permite gerir um fluxo de dados entre o browser e o Serviço Web.

4.2 Serviço de Interacção (Serviço REST)

O serviço de interacção é a parte do projecto que torna efectiva a ligação entre o laboratório remoto e o utilizador final. A VI que serve de ficheiro fonte à criação deste serviço está representada na figura 3.5. A criação de um Serviço Web no Labview consiste primeiramente na criação de "Web Methods", "Web Methods" são ligações que são feitas relativamente às variáveis de entrada e de saída que queremos tornar públicas ao utilizador; deste modo as variáveis passam a ser métodos acessíveis utilizando o protocolo HTTP. O Labview cria Serviços Web baseados na arquitectura restful. O utilizador tem a possibilidade de dar um nome ao serviço Web, definir qual o ficheiro fonte que neste caso será o "filtroweb.vi", assim como definir de que forma este serviço

estará disponível. Existem três formatos para publicar o serviço: HTML, xml e txt. Nesta tese escolheu-se o XML por ser o formato mais comum na Internet e também devido ao facto de ser mais facilmente manipulável por uma aplicação cliente. O parse do XML é feito com recurso ao AJAX que efectua através de rotinas DOM a extracção dos dados necessários ou dos dados que interessa extrair.

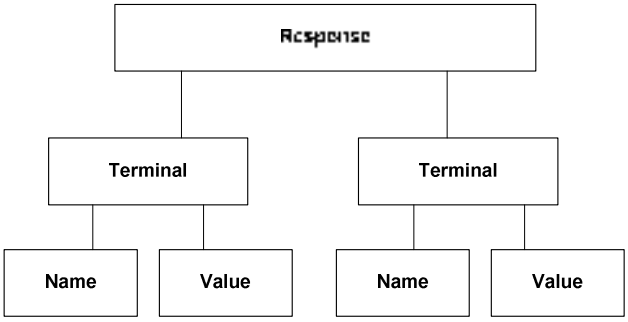

Figura 4.4. Árvore do XML DOM da resposta do servidor

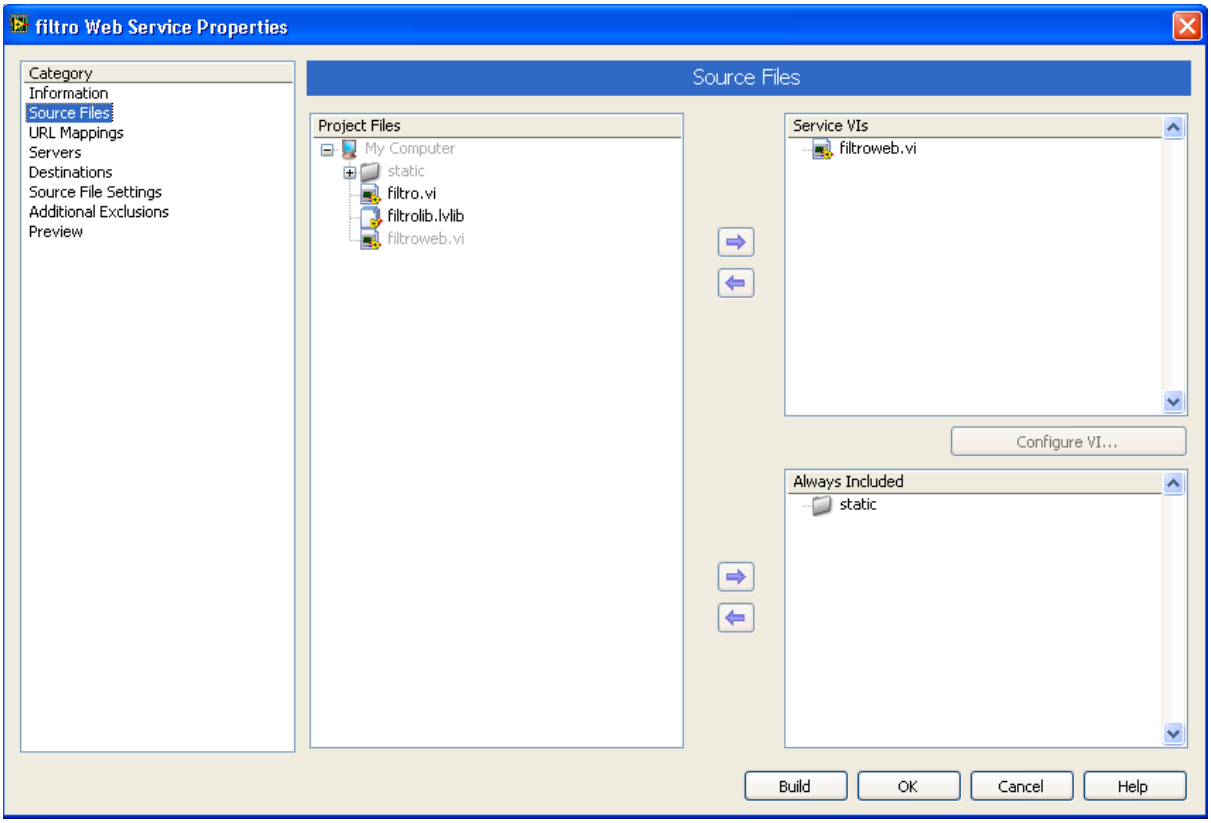

Figura 4.5. Página de configuração de Serviços Web no Labview

O processo de publicação de uma vi como Serviço Web envolve sempre alguns passos importantes. É necessário definir o ficheiro fonte, que por norma é uma vi auxiliar que utiliza as mesmas variáveis da vi principal. O segundo passo consiste em estabelecer o mapeamento das variáveis de entrada ("url mappings"), ou seja criar recursos. Segundo o paradigma do REST, um endereço HTTP é um recurso e neste caso o mapeamento da variável "vin" irá fazer com a mesma transforme-se num recurso que esteja acessível via Internet. Deve também ser definido que primitiva HTTP deve ser usada. Outro passo importante é a criação dos ficheiros HTML que serão usados na interface do utilizador; estes ficheiros terão que ser igualmente mapeados de modo a serem acedidos de forma externa e correrem no servidor Web do Labview aonde corre o Serviço Web e todas as suas componentes ("Web Methods").

Na figura 4.6 encontra-se ilustrada a configuração e o funcionamento do serviço de interacção. O bloco "filtro.vi" não faz parte do serviço ou seja a aplicação principal não se encontra a correr dentro do servidor Web do Labview.

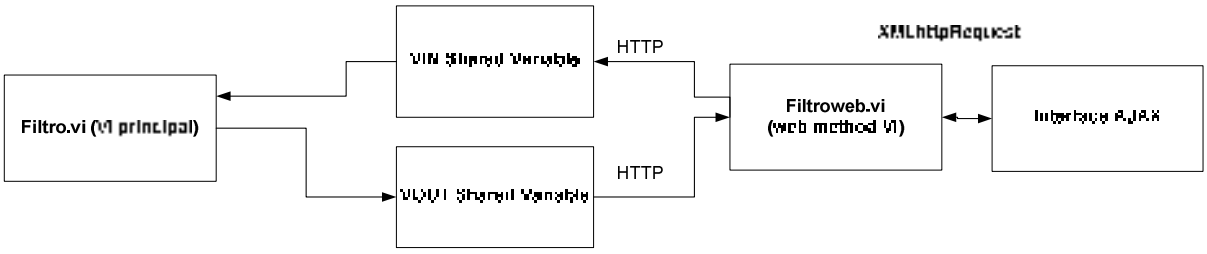

Figura 4.6. Configuração do Serviço Web

Aplicação (VI principal) é totalmente independente do serviço, pois pode funcionar de forma autónoma, a comunicação entre o serviço de interacção e a aplicação "filtro.vi" faz-se por intermédio das variáveis partilhadas, as quais são acessíveis nas duas

aplicações. A comunicação entre a "Web Method VI" e a aplicação cliente é feita recorrendo à primitiva "get" do protocolo HTTP, tanto para receber como para enviar. Esta é pois uma das grandes vantagens da arquitectura REST, que aproveita recursos já existentes, o que torna a comunicação entre a interface cliente e a "Web Method VI" muito simples.

Figura 4.7. Esquema de Comunicação entre o Serviço Web Labview e a aplicação cliente

Na figura 4.7 está ilustrado o processo de comunicação entre as duas aplicações tanto da "Web Method Vi" como da aplicação cliente. Ambas as aplicações são executadas no servidor do Labview (server side). A aplicação cliente é configurada numa directoria virtual especial de nome static; a sua definição é feita no Labview e o seu mapeamento também, aonde lhe é atribuído um endereço HTTP, o que faz dela um recurso REST em formato HTML. É este recurso que vai dar acesso às funcionalidades do serviço ao cliente final, sendo o meio de comunicação entre o cliente final e o núcleo do serviço (filtroweb.vi). Como se pode ver na figura 4.7 o objecto XMLHTTPRequest é o meio que permite ao AJAX efectuar pedidos (requests) ao servidor. O objecto XMLHTTPRequest utiliza as primitivas do protocolo HTTP e permite com isso trocas de informação assíncronas entre o cliente e o servidor.

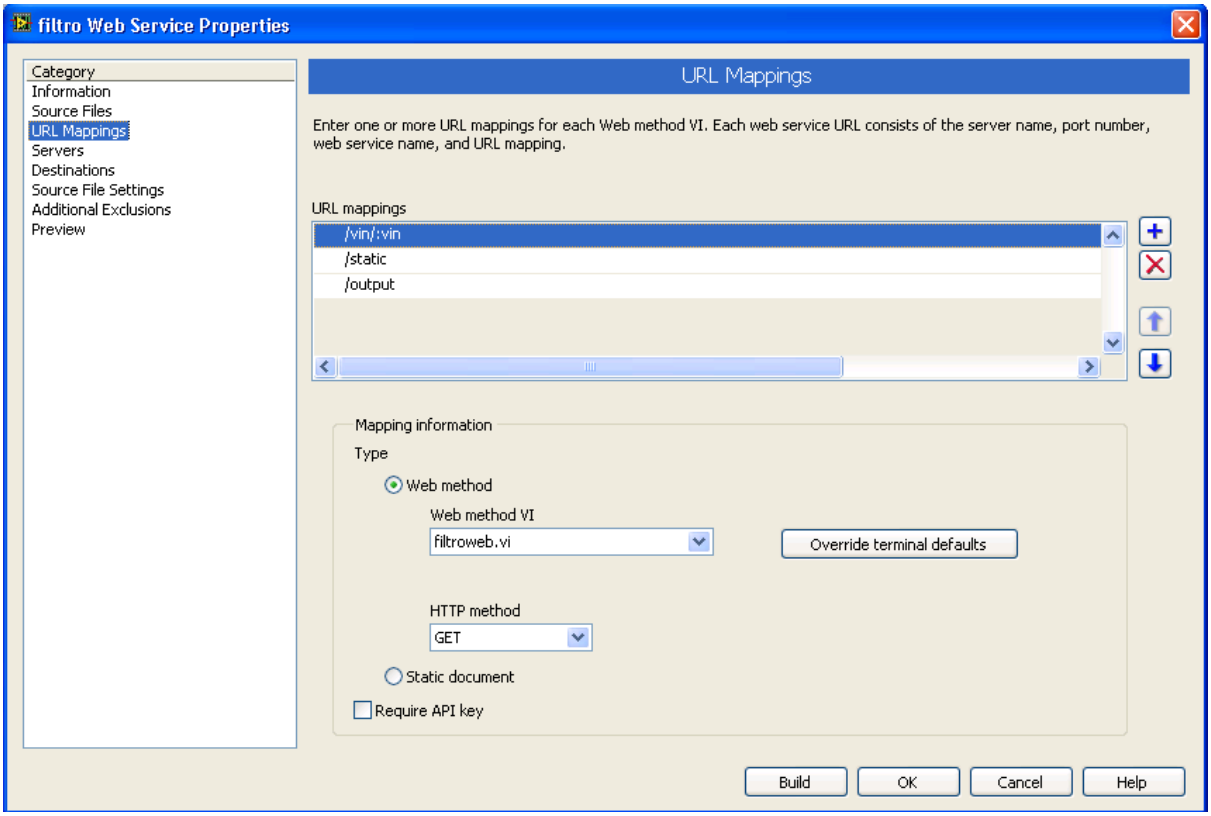

Figura 4.8. Mapeamento da variáveis (vin)

O Labview permite-nos escolher o "Web Method VI" e o "HTTP Method" que neste caso é o "GET". A configuração de páginas HTML e outros documentos que se quer ter acessíveis deverá ser feita seleccionando a opção "Static Document". A figura 4.8 ilustra o mapeamento das variáveis no Labview para o Serviço Restful.

4.3 Aplicação Cliente

Aplicação cliente foi desenvolvida em AJAX e HTML. Todos os seus ficheiros fazem parte do serviço de interacção e são executados do lado do servidor (server side).

Para que o utilizador possa ter acesso a interface é necessário que os ficheiros HTML estejam configurados como "static documents" e sejam mapeados, ou seja é necessário que sejam um recurso do serviço de interacção que possa ser acedido pelo protocolo HTTP. O Labview Web Server executa código javascript ou AJAX; essa particularidade permite que a aplicação cliente possa estar implementada dentro do projecto Labview. Na figura 4.9 verificamos o mapeamento dos ficheiros estáticos.

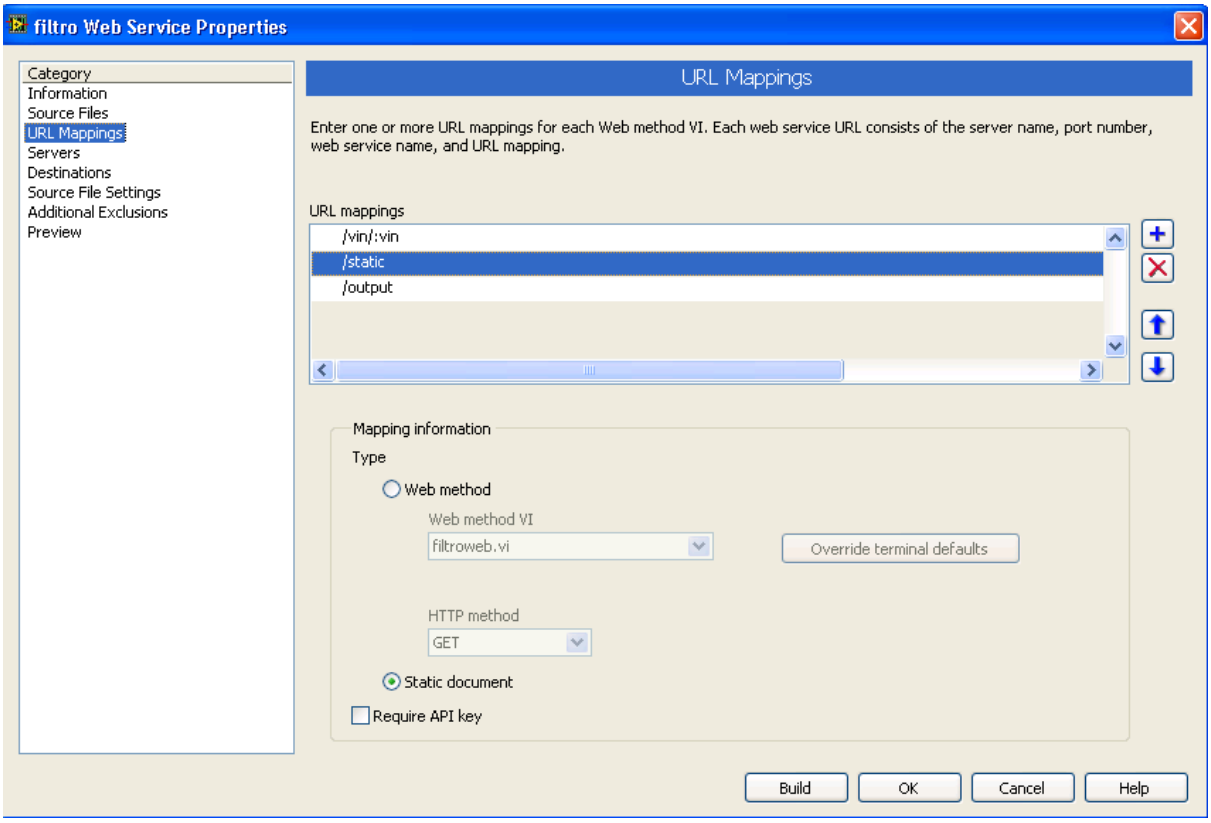

Figura 4.9. Mapeamento dos ficheiros estáticos (HTML)

AJAX significa em sentido literal “Asynchronous Javascript And XML”; não é uma tecnologia, mas antes um conjunto de tecnologias. Podemos considerar que a sua adopção está de certa forma a revolucionar a Internet permitindo que o utilizador possa ter experiencias mais dinâmicas e enriquecedoras. O AJAX é pois constituído pelas seguintes tecnologias:

- XHTML e CSS para apresentação;
- DOM para apresentação dinâmica e interacção;
- XML e XSLT manipulação e troca de dados;
- XMLHTTPRequest download assíncrono de dados;
- Javascript que junta as tecnologias todas

O AJAX faz uso do objecto XMLHTTPRequest para fazer uma ligação ao servidor e fazer o download de dados. Geralmente esses dados estão em formato XML, mas também podem ser em formato TXT ou HTML. O AJAX faz o download de dados de forma assíncrona, ou seja o utilizador não se apercebe do refrescamento do browser, pelo simples facto de que o browser não é refrescado. Ou seja o facto de os browsers já suportarem DHTML e javascript permite que os dados possam ser mudados sem que se faça o recarregamento do browser para actualizar os dados. Na solução em AJAX para projecto do cliente o objecto XMLHTTPRequest fará os downloads de dados de um ficheiro XML.

```
- <Response>
  - <Terminal>
      <Name>vout</Name>
      <Value>0,124442</Value>
    </Terminal>
  - <Terminal>
      <Name>httpRequestID out</Name>
      <Value>363</Value>
    </Terminal>
  </Response>
```

Figura 4.10. Ficheiro XML com valores da variável vout

Na figura 4.10 o ficheiro XML contem os dados da variável “vout” ou seja a saída do processo (filtro de 1ª ordem) quando depois de excitado com um valor de tensão de entrada.

O objecto XMLHTTPRequest é o cérebro do AJAX, portanto a primeira coisa a fazer-se é criar um objecto XMLHTTPRequest. Com esse objecto criado podemos aceder aos recursos que o servidor disponibiliza, nomeadamente ao endereço da variável “vin” que será do tipo HTTP://localhost/filtrowebservice/vin. Tendo acesso ao endereço do recurso podemos executar uma função que receba um valor do utilizador e o aplique à variável; para isso basta colocar o valor que pretendemos directamente numérico directamente no endereço do recurso da seguinte forma HTTP://localhost/filtrowebservice/vin/5. O resultado desta operação vai ser que o ficheiro XML da figura 4.10 vai apresentar o valor correspondente de “vout” para “vin=5” , no entanto como pretendemos que haja um auto refrescamento de dados é necessário usarmos as funções setInterval() e clearInterval que são funções nativas do AJAX, a primeira permite chamar uma função infinitamente e a segunda permite parar a chamada.

A figura 4.11 ilustra o algoritmo feito em AJAX que demonstra como o sistema funciona partindo do momento em que o utilizador insere um valor na página HTML.

O objectivo é ter-se uma ideia de como as transferências de dados funcionam.

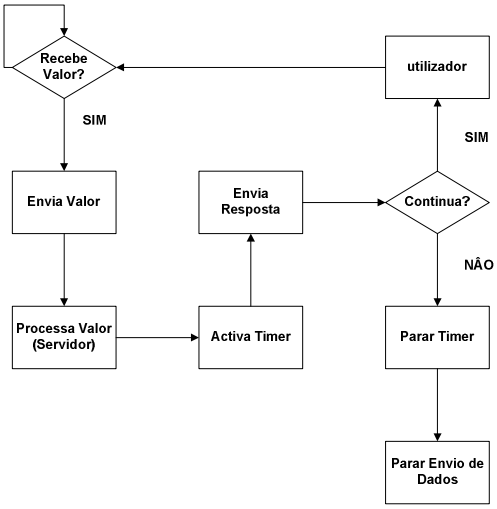

Figura 4.11. Fluxograma da Aplicação Cliente

Neste fluxograma o utilizador submete um valor numérico entre 0 e 5 Volts, este valor é enviado para o servidor que o processa e o sistema reage em conformidade. A aplicação cliente activa um contador (timer) e começa a processar os dados gerados pelo servidor e envia-os para o browser do utilizador. O utilizador pode a qualquer momento parar o envio de informação ou recomeçar o sistema do início.

Seguidamente far-se-á uma descrição das funções usadas no AJAX que foram usadas para a troca de dados entre a aplicação cliente e o servidor.

1. **upload(name,value):** Esta função é responsável pelo envio de dados do cliente para o servidor. É activada quando um valor é detectado numa caixa input text de uma página HTML. Quando um valor é detectado a função envia para o endereço do recurso que pretende alterar (variável vin) através da primitiva HTTP "GET". Utiliza o objecto XMLHTTPRequest para aceder ao servidor.
2. **refreshAjax():** Esta função é responsável faz a chamada a função setInterval() que é uma função nativa do AJAX. Permite transferências dinâmicas de dados
3. **stoprefreshAjax():** Esta função faz a chamada da clearInterval(); o seu objectivo é o de parar a transferência dinâmica de dados
4. **download():** Esta função através da função responseAjax() que utiliza o objecto XMLHTTPRequest, faz uma solicitação de dados ao servidor, permitindo assim

a transferência de dados entre o cliente e o servidor. Utiliza a primitiva "GET" do protocolo HTTP para fazer o download de dados.

5. **responseAjax():** Esta função é chamada pelas funções upload(name,value) e pela função download(). A sua função é utilizar o objecto XMLHTTPRequest para fazer a ligação ao servidor. Para além disso faz o parse do XML que o servidor envia como resposta. No parse do XML extrai apenas o valor de vout ignorando os outros campos que para está experiência não são necessários.
6. **getXMLHTTPRequest():** Esta função é responsável por verificar se o browser do cliente (utilizador) suporta o javascript de forma a determinar se pode ou não correr o AJAX, também é responsável pela criação do objecto XMLHTTPRequest.

O XMLHTTPRequest é um objecto que permite que scripts feitos em javascript possam aceder a um servidor de forma assíncrona utilizando a linguagem HTTP. Está funcionalidade permite que se possa fazer pedidos ao servidor, ou receber respostas ou mesmo fazer a actualização de uma parte da página sem comprometer o normal funcionamento do resto da página. Foi criado pela Microsoft em 1999 e neste momento é um objecto nativo na esmagadora maioria dos browsers.

4.3 Resultados Experimentais

Seguidamente apresentaremos os resultados experimentais com esta solução, na figura 4.12 apresenta-se o painel frontal da aplicação de controlo desenvolvida em Labview (filtro.vi) e na figura 4.13 a página Web que o utilizador terá acesso para controlar e visualizar os dados que vêem do Serviço Web. Os dados na página são actualizados a uma taxa de 1 segundo, no entanto na aplicação a taxa de actualização é maior. Esse facto deve-se a que a utilização de tempos de refrescamento demasiado baixos produz problemas no browser, impedindo que este deixe de funcionar de forma correcta, dai ter sido escolhido o valor de 1 segundo.

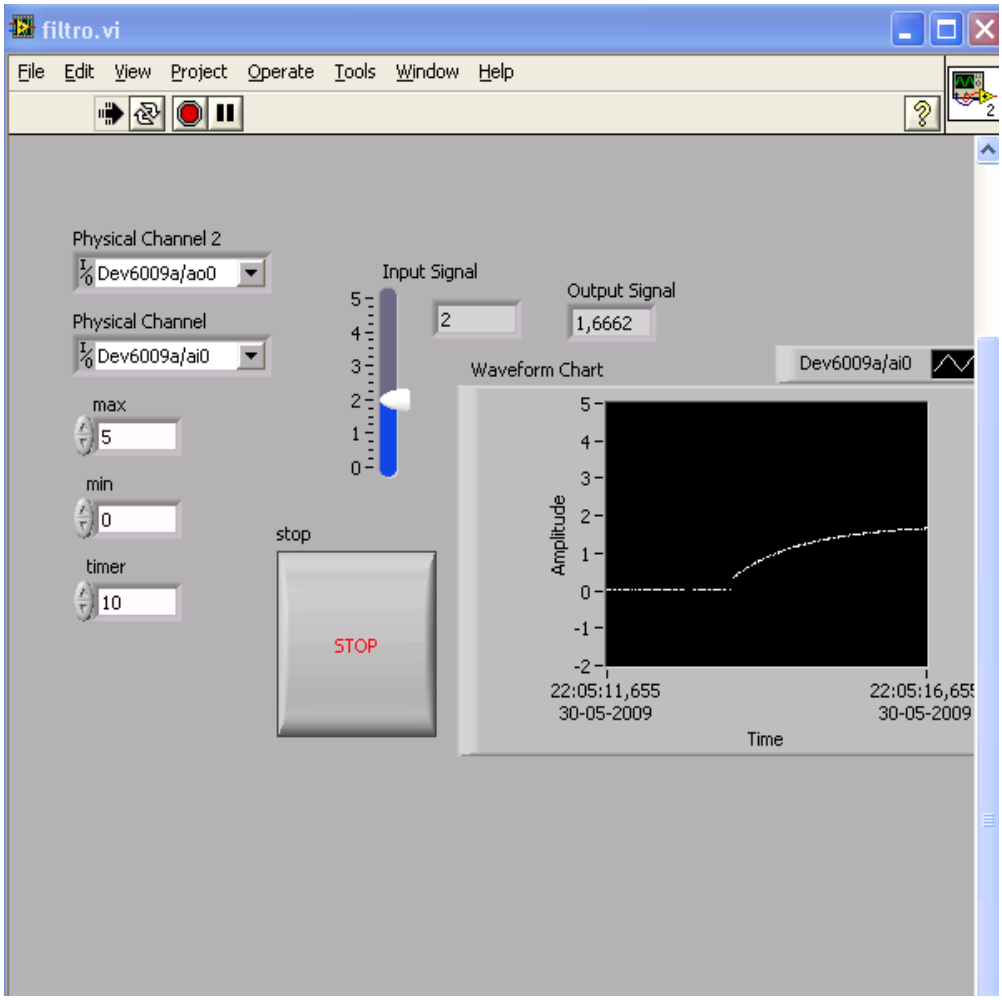

Figura 4.12. Painel de Controlo do VI de controlo

Nesta figura podemos ver que o utilizador pode utilizar a página Web para alterar o valor de input do sistema que leva a resposta a subir ou a descer. O utilizador tal como se estivesse a utilizar o Labview tem também acesso ao valor de saída do sistema "vout", podendo parar a leitura de dados. Desta forma prova-se que a utilização tecnologias comuns em qualquer browser, como o javascript e o AJAX, facilitam o ensino a distância mantendo a riqueza da experiência com troca dinâmica de dados, sem prejudicar o desempenho do sistema do utilizador. O AJAX faz uso de tecnologia que já se encontra presente nos browsers e tira partido disso para criar clientes leves para aplicação em laboratórios remotos. Neste caso particular, o utilizador utilizando o browser introduziu um valor de 2V (vin) como tensão de entrada, o resultado pode ser verificado tanto no browser (figura 4.13) como no painel frontal da VI (figura 4.12). No entanto o valor não é exactamente igual dado que o temporizador do browser exibe um pequeno atraso por questões de optimização, para além disso o próprio sistema (filto de 1ª ordem) exibe ruído ao nível dos conversores A/D e D/A e dos components do circuito.

Nesta página apenas o valor de VOUT é actualizado de forma dinâmica, o browser não sofre problemas de desempenho durante essa actualização uma vez que a página completa não está a ser actualizada mas apenas uma pequena parte. A utilização do objecto XMLHTTPRequest permite uma troca dados dinâmica de dados entre servidor e cliente, sem o refrescamento de toda a página, como já foi referido anteriormente. O sistema implementado permite assim controlar por intermédio de apenas uma página HTML uma experiencia localizada remotamente, sem a utilização de tecnologia adicional. Reduz-se assim a necessidade do utilizador sobrecarregar o seu sistema com a instalação de software adicional para poder aceder à experiência remota.

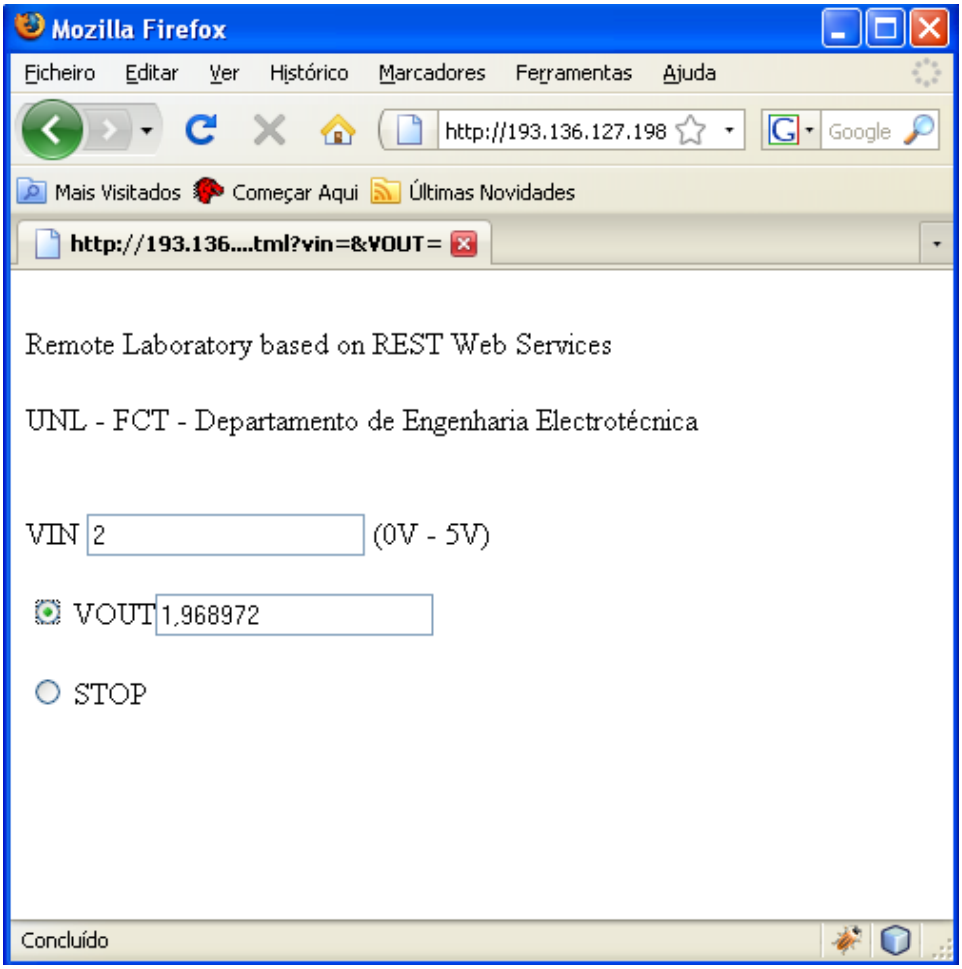

Figura 4.13. Página Web de controlo, o utilizador envia a tensão VIN e lê a resposta VOUT

A experiencia encontra-se no seguinte endereço:

HTTP://193.136.127.198/filtrowebservice/static/index.HTML

4.4 Experimento Filtro Activo de 1ª Ordem

O filtro activo de primeira ordem pode ser usado para simular alguns tipos de sistemas como sistemas de aquecimento, sistemas de nível de líquidos etc.

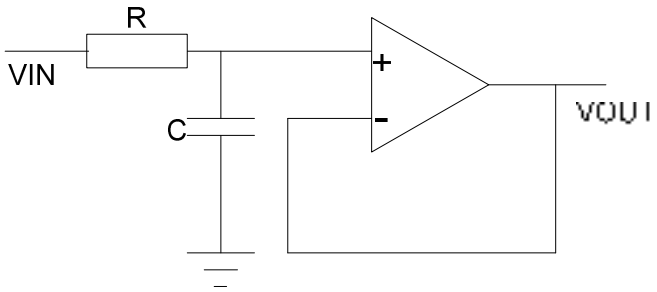

Figura 4.14. Esquema do Filtro Activo de 1ª Ordem

Este sistema (filtro) encontra-se ligado ao sistema Labview através de uma placa de aquisição do tipo NI-USB 6009. É injectado um valor de tensão na entrada "vin" e posteriormente feita a leitura do valor da tensão na saída "vout". Este simples circuito pode ser assim testado remotamente sem a necessidade de se recorrer a presença do utilizador no laboratório, nem a necessidade de se recorrer a instrumentos reais. A instrumentação virtual e os Serviços Web permitem pois a realização de testes, ainda que básicos, a este circuito.

Neste sistema o utilizador pode observar o regime de transitório e o regime permanente que o filtro exibe para valores de tensão de entrada entre 0 e 5V; o valor de ([0:5] V) tem a ver com os limites de tensão da placa NI-USB 6009. A variação que é possível de seguir com o Labview no laboratório pode ser feita usando o browser que vai actualizando sequencialmente os valores que o Labview vai publicando no Serviço Web.

4.5 Conclusões

Neste capítulo foi apresentada a solução desenvolvida para laboratórios remotos baseada em conformidade com o tema da tese. Apresentou-se também um cliente leve baseado em AJAX e aplicação servidora em Labview 8.6. No capítulo que se segue serão feitas as conclusões finais e indicados alguns apontadores para trabalho futuro.

5 Conclusões e Trabalho Futuro

5.1 Conclusões Finais

Nesta dissertação foi proposta uma nova arquitectura para laboratórios remotos, utilizando Labview no lado do servidor e um cliente leve baseado em browser no lado do cliente. Foram utilizadas tecnologias recentes como os Serviços Web, e outras recuperadas como o AJAX, para a implementação de laboratórios remotos, fundamentalmente com vista a dois objectivos principais:

- Redução de custos globais
- Utilização optimizada da Internet por intermédio do browser

A maioria dos grandes fabricantes de software científico como a National Instruments e a MathWorks têm vindo a transformar os seus produtos de modo a que os mesmos se possam rapidamente adaptar às transformações que os sistemas de informação estão constantemente a sofrer. Particularmente no caso da National Instruments e da sua principal aplicação o Labview, assistimos na sua última versão a disponibilização de suporte aos Serviços Web. Essa modificação no software Labview permite desde logo aproveitar mais está capacidade do pacote deste software, permitindo a criação de soluções em linha (“online”) sem que seja necessário que do lado do cliente exista o pesado pluggin da National necessário para que o utilizador possa aceder as experiências do Labview sem ter instalado na sua máquina o software da National. Infelizmente essa solução não faz uma utilização optimizada daquilo que já está disponível em termos de tecnologia Internet. Esse facto torna a solução desenvolvida neste trabalho mais atractiva e mais simples para o utilizador, uma vez que fazendo uso de um cliente AJAX para aceder aos Serviços Web criados pelo Labview o utilizador apenas precisa de utilizar aquilo que já tem disponível na sua máquina, o browser.

Podemos então afirmar que do ponto de vista dos custos está solução ou soluções que sigam o mesmo paradigma deste trabalho permitem oferecer soluções mais optimizadas e eficientes do ponto de vista da utilização da Internet, por outro lado o facto de o AJAX não ser uma tecnologia proprietária permite desenvolver soluções nas quais se evita despender demasiados custos ao nível do software e ao mesmo tempo oferecer soluções que permitam aos clientes utilizarem pouca largura de banda, não perdendo contudo a riqueza da experiência.

Com esta arquitectura foram alcançados os pressupostos iniciais de encontrar uma solução que permitisse o uso de um cliente simples e leve para laboratórios remotos.

5.2 Trabalho Futuro

Futuramente esta solução será transformada numa solução mais escalável de modo a que possam coexistir mais do que uma VI a controlar experiências diferentes e consequentemente ser criado uma opção multi-serviços em que cada experiência pudesse oferecer serviços múltiplos aos seus utilizadores. De uma maneira geral cada experiência seria simultaneamente um serviço (serviço composto) e ao mesmo tempo ofereceria serviços ao utilizador.

Outro factor a ter em conta é a de fornecer uma gestão de acesso de modo a resolver os casos de utilização de serviços de modo concorrente, o sistema deveria detectar se um determinado serviço está a ser utilizado e dar ao utilizador a possibilidade de escolher outro serviço ou de aguardar um tempo determinado.

Poderia igualmente ser incluída nesta solução a utilização do pluggin da adobe o SVG que se está a tornar rapidamente um padrão nos bowsers (firefox). O SVG permite a utilização de elementos gráficos nas páginas Web sem redução da eficiência.

Anexo A – Placa NI-USB6009

A placa de aquisição NI-USB6009 é uma placa de aquisição fabricada pela National Instruments.

Figura A.1. Placa NI-USB 6009

Na figura A.1 está representada a da placa e a placa NI-USB 6009 (ou 6008). Está placa é compatível com os seguintes sistemas operativos:

- Windows Vista, XP, 2000;
- Mac OS X;
- Linux (Mandriva, Suse e RedHat);
- Windows Mobile;
- Windows CE;

A placa USB-6009 suporta de forma nativa toda a família da Suite Labview porém suporta também rotinas desenvolvidas em C#, Visual Basic .Net e ANSI C/C++. As suas principais características são:

- 8 Entradas analógias a 14 bits e com a velocidade de 48KS/s;

- 2 Saídas analógicas a 12 bits;
- Contador de 5MHz a 32 bits;
- 12 Linhas TTL digitais input/output;
- Trigger digital;

Anexo B – SVG (Scalable Vector Graphics)

SVG é uma tecnologia emergente que poderá ou não revolucionar os gráficos na Internet. O SVG é uma linguagem que define gráficos bidimensionais. O SVG foi recomendado em 2003 pelo W3C e em alguns browsers já é um standard como o firefox. O SVG foi criado por um conjunto de empresas entre as quais se destacam as seguintes: Sun Microsystems, Adobe, Apple, IBM e Kodak. O SVG tem as seguintes características:

- Os gráficos SVG são definidos em formato XML, o que permite uma manipulação fácil através dos editores e parsers de XML.
- As imagens SVG são escaláveis
- Os SVG contêm fontes que permitem a preservação tanto de imagem como de texto
- Contêm elementos declarativos animados
- Suporta um multi-espaço de nomes em XML

Para se poder correr aplicações que façam uso do SVG é necessário a instalação de um pluggin localizado no seguinte site HTTP://www.adobe.com/svg/viewer/install/main.HTML

Anexo C – O REST no mundo real

Existem vários exemplos de aplicação de serviços REST na Internet. Muitas aplicações Web providenciam interfaces que podem ser utilizadas por programadores para implementar soluções por cima destas interfaces. Como exemplos de serviços REST disponível na Internet temos os seguintes:

- BBC news feed;
- Yahoo news search;
- Yahoo maps;
- Yahoo local search;
- Earthquakes feed;

Actualmente os serviços REST possuem muitas formas. Alguns desses serviços possuem API's próprias como a Flickr API, Amazon API e a Yahoo API, que já definem os formatos de dados de entrada e saída e a informação sobre o URL do recurso.

Para ser consumir um destes serviços é necessário cumprir os seguintes passos:

- Descobrir os parâmetros de entrada e o formato desses parâmetros;
- Descobrir o endereço URL do recurso que se quer consumir e o verbo HTTP que o serviço espera que o programador use;
- Descobrir o formato da resposta e descobrir como processar a resposta esperada de forma a recolher a informação pretendida na resposta;
- Consultar os termos de uso do serviço;

Anexo D - DOM

O DOM (“Document Object Model”)é uma API que define um conjunto de objectos com as suas propriedades e características. O DOM foi projectado para suportar tanto o HTML como o XML. O DOM é um conjunto de recomendações e apesar de ser muitas vezes referido como um grande standard encontra-se dividido em 4 níveis. Nivel 0,1,2 e 3. Uma breve descrição dos níveis é feita seguidamente:

1 – Nível 0: não existe como recomendação do W3C é usado para descrever as características presentes na versão 3 dos browsers Netscape e Internet Explorer

2 – Nível 1: é o núcleo da API, recomendado pelo W3C foi especificado para XML e HTML e contem extensões HTML.

3 – Nível 2: adiciona extensões específicas ao HTML e ao XML que suportam por exemplo eventos.

6 - Nível 3: adiciona extensões ao núcleo da API e manipulação de eventos.

Para se poder extrair um determinado elemento usando DOM deve fazer-se o seguinte é necessário fazer o seguinte:

Childnodes – permite extrair todos os nós abaixo do elemento corrente

parentNode - extrai o parente directo do elemento corrente

nextSibiling/previousSibiling – extrai o próximo ou o nó seguinte

firstchild/lastchild – extrai o primeiro ou o ultimo nó

Referências

Coito, Fernando, L. Brito Palma, *A Remote Laboratory Environment for Blended Learning*, PTLIE Workshop – Pervasive Technologies in E/M Learning and Internet Based Experiences, 1st ACM International Conference on Pervasive Technologies Related to Assistive Environments (PETRA 2008), July 16 – 18, Athens – Greece

Ngolo, Márcio, Fernando Coito, L. Brito Palma, Luís Gomes, Anikó Costa, *Architecute for Remote Laboratories Based on REST Web Services,* ICELIE09 3rd International Conference on e-Learning in Industrial Electronics – ICELIE09, November 3 – 5, Porto – Portugal (accepted)

Baccigalupi, A. , C. de Capua, A. Liccardo, *Overview on Developing of Remote Teaching Laboratory: From Labview to Web Services,* IMTC2006, Instrumentation and Measurement Technology Conference, April 24 – 27, Sorrento – Italy

Páginas de Internet:

HTTP://www.w3c.org

HTTP://ieee.org

HTTP://www.ni.com

HTTP://forum.ni.com

Outras Fontes (Livros, Teses Mestrado e Doutoramento):

Leonard Richardson and & Sam Ruby, *Restful Web Services*

Samisa Abeysinghe, *RESTful PHP Web Services*

Steven Holzener, *AJAX a Beginner's Guide*

Luís Gomes and Javier Garcia-Zubía, *Advanced on Remote Laboratories and e-learning experiences*

S.Sumati, P. Sureka, Labview Based Advanced Instrumentation and Control

Fielding, Roy Thomas (2000), *Architectural Styles and the Design of Network-based Software Architectures, Tese de Doutoramento,* California, USA

Coelho, Paulo (2006) Uma *Arquitectura Orientada a Serviços Para Laboratórios Remotos,* Tese de Mestrado, São Paulo, Brasil